PENGEMBANGAN APLIKASI KATALOG DAN PEMESANAN PRODUK KEBUTUHAN DAPUR BERBASIS ANDROID MENGGUNAKAN METODE MOBILE-D

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Arya Yudha Mahendra
NIM: 135150200111092

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PENGEMBANGAN APLIKASI KATALOG DAN PEMESANAN PRODUK KEBUTUHAN DAPUR BERBASIS ANDROID MENGGUNAKAN METODE MOBILE-D

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Arya Yudha Mahendra
NIM: 135150200111092

Dosen Pembimbing I	Dosen Pembimbing II
Adam Hendra Brata, S.Kom., M.T., M.Sc.	Komang Candra Brata, S.Kom., M.T., M.Sc.
NIK: 2016079001051001	NIK: 2016078907111001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D
NIP: 197105182003121001

IDENTITAS TIM PENGUJI

No.	Nama Dosen Penguji	NIK
1.	Agi Putra Kharisma, S.T, M.T (penguji ke I / ketua majelis)	201304 860430 1 001
2.	Mahardeka Tri Ananta, S.Kom., M.T., M.Sc. (penguji ke II)	201607 891204 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 4 Juni 2018

Arya Yudha Mahendra

NIM: 135150200111092

DAFTAR RIWAYAT HIDUP

Data Pribadi

Nama	: Arya Yudha Mahendra
Jenis Kelamin	: Laki – laki
Tempat/Tgl Lahir	: Martapura OKU, 3 Mei 1994
Agama	: Islam
Status	: Belum Menikah
Alamat	: Jl. Tapak Siring IV/14C, Klojen Kota Malang
Telp / Hp	: 082226868403

Latar Belakang Pendidikan

A. Pendidikan Formal
   1. Taman Kanak-Kanak Muslimat NU Ngebruk, Kab. Malang
   2. Sekolah Dasar Negeri Lowokwaru 3 Kota Malang
   3. Sekolah Menengah Pertama Negeri 3 Kota Malang
   4. Sekolah Menengah Atas Negeri 8 Kota Malang

B. Pendidikan Non Formal

Pengalaman Kerja

1. UKM Street Matcha
2. Ketua Kelompok Program Kreativitas Mahasiswa oleh Kemenristek Dikti
3. Web Programmer Sistem Monitoring Rencana Kerja pada PT. PLN TJBTB

Pengalaman Organisasi

1. 2013, Anggota LSO POROS Fakultas Ilmu Komputer Universitas Brawijaya
2. 2014, Anggota Divisi Pendamping Pengenalan Kehidupan Kampus Mahasiswa Baru. FILKOM - Universitas Brawijaya
3. 2015, Anggota Unit Aktivitas Mahasiswa RKIM Universitas Brawijaya

Skripsi Ini Saya
Persembahkan Untuk Keluarga
Dan Para Sahabat...
Terimakasih Sudah Selalu
Memberikan Dukungan Semangat
Dan Motivasi Untuk Penulis.

KATA PENGANTAR

Puji syukur kehadirat Tuhan YME yang telah melimpahkan rahmat-Nya sehingga laporan skripsi yang berjudul "Pengembangan Aplikasi Katalog Dan Pemesanan Produk Kebutuhan Dapur Berbasis Android Menggunakan Metode MOBILE-D" ini dapat terselesaikan.

Penulis menyadari bahwa skripsi ini tidak akan berhasil tanpa bantuan dari beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih kepada:

1. Mama Tri Setyaningsih, atas perhatian, semangat dan kesabarannya dalam membesarkan dan mendidik serta merawat keseharian penulis.
2. Papa Agus Asmarayana, atas perhatian, arahan dan kesabarannya dalam membesarkan dan mendidik penulis.
3. Mas Aditya Indra Mahendra, atas perhatian, arahan dan kritik yang membuat penulis bersemangat untuk menyelesaikan skripsi ini.
4. Bapak Adam Hendra Brata, S.Kom., M.T., M.Sc. dan Bapak Komang Candra Brata, S.Kom., M.T., M.Sc. selaku dosen pembimbing skripsi yang telah dengan sabar membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
5. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D. selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
6. Seluruh teman – teman, khususnya Meita Putri yang senantiasa membantu, memberikan kritik, saran dan solusi kepada penulis selama penulis menyusun skripsi ini.
7. Seluruh responden baik dari sisi konsumen maupun penjual sayur keliling yang telah bersedia meluangkan waktunya untuk memberikan jawaban atas pertanyaan – pertanyaan penulis serta memberikan masukan yang sangat berharga terhadap pengembangan aplikasi Mlijo ini.
8. Semua pihak dan teman-teman lainnya ikut membantu dalam proses penyelesaian skripsi dan selalu memberikan semangat kepada penulis.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak kekurangan, sehingga saran dan kritik yang membangun sangat penulis harapkan. Akhir kata penulis berharap skripsi ini dapat membawa manfaat bagi semua pihak yang menggunakannya.

Malang, 4 Juni 2018

Penulis

ABSTRAK

Pertumbuhan pengguna internet setiap tahunnya terus bertambah pesat yang juga di tunjang dengan pertumbuhan pengguna perangkat penunjangnya. Perangkat aplikasi bergerak atau mobile merupakan pendukung utama pertambahan pengguna internet yang pada wilayah Indonesia sendiri mencapai angka 47% dari seluruh jumlah penduduk. Dari data tersebut sekitar 46% menggunakannya untuk mengunjungi situs perdagangan online. Android menjadi paling dominan di antara sistem operasi lainnya dengan 77% pengguna perangkat mobile menggunakan android. Hal ini tentu sangat bermanfaat terutama bagi warga Kota Malang yang 34% warganya berprofesi sebagai pedagang. Oleh karna itu aplikasi penunjang untuk melakukan perdagangan tersebut sangat di perlukan pedagang maupun konsumen untuk memenuhi proses bisnis jual beli konsumen dan penjual. Penelitian ini akan meneliti pengembangan aplikasi jual beli antara penjual dan konsumen dengan mengedepankan kemudahan pengoperasian aplikasi. Metode mobile-d diadopsi karena telah mendapatkan sertifikat CMMI level 2 dan memiliki alur pengembangan yang cocok digunakan untuk pengguna yang tidak berada langsung disisi pengembang(*off-site customer*). Disamping itu Mobile-d bersifat iteratif dapat memenuhi kebutuhan yang berubah ubah yang mana cocok untuk menggali kebutuhan langsung dari pengguna aplikasi(*end-user*).

Dalam pengumpulan data menggunakan cara wawancara dan kuisioner. Berdasarkan hasil penelitian yang dilakukan, penilaian pengguna terhadap aplikasi dengan nilai rata-rata skor *SUS* sisi konsumen 84.37 dan pada sisi penjual 72,5, maka dapat disimpulkan bahwa Sistem Katalog Dan Pemesanan Produk Kebutuhan Dapur Berbasis Android mudah digunakan dan dapat diterima oleh pengguna. Angka tersebut membuktikan jika aplikasi memiliki kegunaan, kemudahan, dan kepuasan yang cukup baik bagi pengguna.

Kata kunci: android, mobile-d, *mobile*

ABSTRACT

Each year, Internet users are growth rapidly. Those are also supported with the growth of the internet supporting device user. Mobile application device is a major booster on the increases of internet users in Indonesia, it's alone reaches 47% of the total population. From those data, about 46% use internet to visit online market place. Android becomes the most popular operating systems among others, with 77% of mobile device users use it. It is certainly very useful for residents of Malang City which 34% of population are sellers or merchants. Therefore, a supporting applications that facilitate transaction processes is necessary both for the sellers and consumers. This research will examine application development that can facilate transaction between sellers and consumers by emphasizing the ease of operation. The mobile-d method is adopted because it has obtained a level 2 CMMI certificate and has a development flow that is suitable for off-site customer. In addition, Mobile-d is iterative to meet changing of needs which are suitable for exploring the immediate needs of end-user users.

To collect data, reseacher use interviews and questionnaires. Based on the results of research conducted, the assessment of users of the application has an average score SUS value of 84,37 on the consumer side and 72,5 on the seller side. Therefore, it can be concluded that Catalog System And Kitchen Supplies Android-Based easy to use and acceptable to users. The number prove if the app has a usefulness, ease, and purity is good enough for the user.

Keyword: android, mobile-d, mobile

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Perkembangan teknologi dewasa ini semakin mengalami kemajuan yang pesat, khususnya dibidang teknologi informasi. Pemanfaatannya dalam berbagai bidang bertujuan untuk membantu aktifitas manusia. Hal ini ditunjukan dengan penggunaan perangkat komputer, perangkat aplikasi bergerak atau *mobile*, berbagai jenis aplikasi pendukung dan internet sebagai sarana jembatan pertukaran informasi. Dikutip dari *wearesocial.com* pengguna perangkat aplikasi bergerak yang terkoneksi internet atau pengguna aktif *mobile* internet pada 2017 tercatat sejumlah 3.448 miliyar orang atau 46% dari jumlah penduduk, yang mana mengalami pertumbuhan rata-rata sekitar 50% pengguna secara global. Indonesia sendiri tercatat sebanyak 123,3juta orang atau 47% penduduk adalah pengguna *mobile* internet. Untuk penggunaannya sendiri rata – rata penduduk Indonesia menghabiskan waktu 3 - 4jam dengan sebaran umur 16 – 64 tahun, dimana sekitar 48% menggunakan untuk mencari produk atau jasa *online*, 46% menggunakan untuk mengunjungi situs perdagangan *online* dan 33% melakukan pembelian *online* via *mobile* internet (WeareSocial, 2017). Dari pengguna tersebut, 76,67% merupakan pengguna sistem operasi *Android*, 3,38% iOS, dan 7,73% tidak teridentifikasi (Statcounter, 2017). Dari data tersebut, 65% pengguna berada di pulau Jawa dimana penulis melakukan penelitian ini (APJII, 2016).

Dari data tersebut, penulis kemudian melakukan survei kepada masyarakat pengguna perangkat *mobile* di salah satu kota yang berada di Kota Malang Jawa Timur. Survei dilakukan terhadap 40 orang dari segala sebaran umur dan jenis kelamin, terhadap kebiasaannya dalam berbelanja produk kebutuhan dapur untuk keperluan sehari - hari. Penulis juga melakukan survei terhadap pemasok kebutuhan, yang pada penelitian ini merupakan penjual sayur keliling berdasarkan area berjualannya sebanyak 5 responden. Penulis mengambil topik penjual sayur dikarenakan sekitar 129ribu atau 34% penduduk Kota Malang beprofesi sebagai pedagang, dimana penjual sayur keliling termasuk dalam kategori tersebut (Badan Pusat Statistik, 2016). Survei dilakukan dengan tujuan mengetahui bagaimana proses jual beli yang ada saat ini antara penjual sayur keliling dengan konsumen berikut sistem yang digunakan. Survei berisikan pertanyaan faktor – faktor yang mempengaruhi konsumen dalam berbelanja dan kepuasan terhadap kondisi saat ini. Sedangkan untuk penjual, survei berisikan kendala – kendala yang dialami dalam berdagang, seperti faktor dalam menentukan barang dagangan, faktor pembeli tetap, faktor jam operasional dan lain sebagainya. Dan dari hasil survei tersebut menunjukan hanya 27,5% responden merasa puas dengan kondisi jual beli saat ini. Mereka juga mengatakan perlu adanya aplikasi untuk menunjang proses jual beli tersebut, yaitu sebanyak 85% responden. Pada penjual 100% responden mengatakan mereka pernah merugi dalam berjualan dengan *range* 1-5%. Kemudian sekitar 100% dari penjual tersebut juga mengungkapkan berminat terhadap aplikasi yang dapat membantu mereka dalam proses berjualan.

Berdasarkan data tersebut, penulis mencoba membuat suatu aplikasi berbasis *android* untuk memenuhi kebutuhan masyarakat dengan mobilitas tinggi seperti penjual sayur sehingga mudah dibawa kemanapun mereka berpindah pindah tempat. Dengan menggunakan pendekatan analisis kemudahan pengguna dalam memahami suatu aplikasi, penulis menggunakan metode *Mobile-D* yang merupakan turunan dari metode *agile,* untuk menghasilkan suatu produk yang berkualitas. Menurut (Leau, et al., 2012), pendekatan metode *agile* merupakan pendekatan yang tepat untuk proses pembangunan aplikasi agar lebih memperpendek siklus pengembangan, mengurangi tingkat *bug*, dan untuk mengakomodasi perubahan kebutuhan selama proses pembangunan sehingga tetap memuaskan pelanggan. Terdapat beberapa metode *agile* untuk pengembangan berbasis *mobile*, seperti Mobile-D, RaPiD7, Hybrid *Method*ology Enggineering, MASAM, dan SleSS (Flora & Chande, 2013). Dari beberapa metode tersebut, penulis memutuskan untuk menggunakan metode Mobile-D, karena metode tersebut telah telah diuji secara empiris dan dikembangkan dalam industri pengembangan *mobile* oleh suatu laboratorium di VTT, *The Technical Research Centre Finlandia*. Selain itu Mobile-D telah memperoleh sertifikat CMMI level 2 yang mana metode lain yang telah disebutkan sebelumnya tidak memilikinya (VTT Electronics, 2004).

Metode pendekatan dengan Mobile-D didasarkan pada Extreme Programming XP (*development practices*), Crystal *method*ologies (*method calability*), dan Rational Unified Process(*life-cycle coverage*). Pendekatan Mobile-D dioptimalkan untuk tim kurang dari sepuluh orang pengembang yang bertujuan memaksimalkan pengembangan aplikasi *mobile* dalam jangka waktu yang singkat (Flora & Chande, 2013). Mobile-D terdiri dari lima fase, yaitu: *Explore, Initialize, Productionize, Stabilize,* dan *System Test & Fix*. Mobile-D telah diterapkan dalam proyek-proyek pembangunan perangkat lunak, dan beberapa keuntungan telah diteliti, seperti peningkatan kemajuan visibilitas, pendeteksian dini dan perbaikan masalah teknis, kepadatan cacat yang rendah pada produk akhir, dan kemajuan konstan dalam pembangunan (Sapataru, 2010). Diharapkan dengan penggunaan metode Mobile-D penulis dapat mengembangkan aplikasi Mlijo ini dengan baik sehingga dapat digunakan oleh penjual sayur dan konsumen dengan mudah.

1.2 Rumusan masalah

1. Bagaimana hasil dari pengembangan sistem aplikasi Mlijo pada tahap analisis dengan menggunakan metode Mobile-D?
2. Bagaimana hasil dari pengembangan sistem aplikasi Mlijo pada tahap perancangan dengan menggunakan metode Mobile-D?
3. Bagaimana hasil dari pengembangan sistem aplikasi Mlijo pada tahap implementasi dengan menggunakan metode Mobile-D?
4. Bagaimana hasil dari pengembangan sistem aplikasi Mlijo pada tahap pengujian dengan menggunakan metode Mobile-D?
5. Apakah aplikasi Mlijo dapat dengan mudah digunakan oleh pengguna?

1.3 Tujuan

Tujuan umum:

Mengembangkan aplikasi perangkat lunak bergerak Mlijo agar dapat mempermudah penjual sayur keliling dalam berjualan sehari hari dan konsumen dalam mencari kebutuhan dapur.

Tujuan khusus:

1. Menerapkan metode Mobile-D ke dalam pembangunan perangkat lunak bergerak Mlijo.
2. Membuat aplikasi Mlijo sesuai dengan metode Mobile-D agar dihasilkan aplikasi yang baik, namun dalam waktu pengembangan yang singkat.
3. Mengimplementasikan rancangan yang telah dibuat kedalam kode program berbasis *platform android*.

1.4 Manfaat

Manfaat dari penelitian ini secara umum bagi penjual sayur agar dapat menjual dan berbelanja barang dagangannya sesuai dengan kebutuhan konsumen, sehingga dapat mengurangi tingkat kerugian dalam berjualan. Sedangkan bagi konsumen penjual sayur adalah dapat dengan mudah dalam melakukan belanja kebutuhan dapur dan juga dapat menghemat waktu yang digunakan dalam melakukan belanja kebutuhan dapur setiap harinya. Dan secara khusus bagi penulis, penelitian ini diharapkan dapat membuat penulis semakin memahami dalam melakukan proses pembangunan suatu proyek aplikasi perangkat lunak.

1.5 Batasan masalah

Berdasarkan permasalahan yang telah di jelaskan pada pendahuluan diatas, maka dibuat batasan masalah dalam penelitian ini sebagai berikut:

1. Aplikasi ini hanya terbatas untuk wilayah Kota Malang.
2. Penelitian ini tidak menggunakan prinsip *agile pair programming* pada fase implementasi kode program.
3. Peran pengguna pada penelitian ini hanya terbatas pada penggalian kebutuhan dan *feedback* UI aplikasi.
4. Penelitian ini tidak menerapkan tahapan yang ada pada metode Mobile-D yang berkaitan dengan manajemen tim.
5. Penelitian ini berfokus pada proses pengembangan perangkat lunak dengan menggunakan metode Mobile-D.
6. Penelitian ini melakukan iterasi sebanyak dua kali saja.

1.6 Sistematika pembahasan

Dalam penyusunan makalah skripsi ini terdapat penulisan yang terstruktur sebagai berikut:

BAB 1 PENDAHULUAN

Menguraikan masalah umum yang terkait dengan penelitian bersistematis dan berkesinambungan. Penulisan makalah dimulai dari latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika pembahasan tentang pengembangan aplikasi perangkat lunak bergerak MLIJO dengan metode Mobile-D berbasis *mobile* pada *platform* android.

BAB 2 LANDASAN KEPUSTAKAAN

Membahas kajian pustaka dan dasar teori berdasarkan referensi-referensi yang saling berkaitan dan mendukung satu sama lain dalam pengembangan aplikasi perangkat lunak bergerak MLIJO dengan metode Mobile-D.

BAB 3 METODOLOGI

Menguraikan langkah-langkah penelitian secara sistematik dalam menerapkan metode Mobile-D dalam pembangunan perangkat lunak. Bab ini yang terdiri dari studi literatur, Analisis Kebutuhan, Perancangan Sistem, Implementasi Sistem, Pengujian dan Analisis Sistem serta Penarikan Kesimpulan.

BAB 4 REKAYASA KEBUTUHAN

Membahas analisis kebutuhan sistem dan juga perancangan sistem yang akan dibangun serta perancangan tampilan pengguna sistem beserta alur jalannya program yang akan di bangun.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Membahas bagaimana pengimplementasian rancangan sistem yang telah dibuat pada bab sebelumnya ke dalam kode program pada platform yang telah dipilih sebelumnya.

BAB 6 PENGUJIAN

Membahas tentang pengujian sistem yang telah dibangun sebelumnya sesuai dengan metode yang digunakan untuk memperoleh hasil yang diharapkan dan mengetahui tingkat kehandalan suatu sistem.

BAB 7 PENUTUP

Bab ini merupakan bab terakhir yang digunakan untuk menguraikan kesimpulan yang diperoleh dari pembangunan perangkat lunak bergerak. Bab ini juga memiliki subbab kritik dan menyertakan saran yang dapat digunakan untuk pengembangan selanjutnya agar memiliki hasil akurasi yang lebih baik.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab landasan kepustakan ini berisi tentang kajian pustaka terhadap penelitian-penelitian terkait yang berhubungan dengan penelitian dan pembahasan tentang dasar teori yang dapat mendukung dalam penyusunan penelitian ini.

2.1 Kajian Pustaka

Pada bagian kajian pustaka akan menampilkan penelitian terdahulu yang berhubungan dengan penelitian ini dengan menggunakan metode pengembangan Mobile-D. Selain itu juga akan dijelaskan kajian pustaka lainnya yang berhubungan dengan metode pengembangan sistem berbasis aplikasi *mobile*. Kajian pustaka pada penelitian ini dapat dilihat pada tabel 2.1 berikut :

Tabel 2.1 Kajian Pustaka Penelitian Terkait

No.	Judul Penelitian	Topik Penelitian	Objek Penelitian	Keunikan Penelitian
1.	*Agile Software Development Processes for Mobile Systems: Accomplishment, Evidence and Evolution* (Corral, et al., 2013)	Rekayasa Perangkat Lunak	Metode pengembangan perangkat lunak menggunakan metode agile untuk sistem mobile	Memaparkan tinjauan tentang proses pengembangan perangkat lunak Agile untuk aplikasi *mobile* dan implementasinya.
2.	*Mobile-D: An Agile approach for mobile application development* (Abrahamsson, et al., 2004)	Rekayasa Perangkat Lunak	Metode Pengembangan untuk sistem berbasis *mobile*	Memaparkan sebuah metode pengembangan baru dalam pengembangan aplikasi perangkat lunak berbasis *mobile* yaitu Mobile-D
3.	*Agile Development Methods for Mobile Applications* (Sapataru, 2010)	Pengembangan perangkat lunak berbasis *mobile*	Pengembangan metode Mobile-D untuk sistem berbasis *mobile*	Memaparkan penggunaan metode Mobile-D untuk

		dengan metode Mobile-D		pengembangan aplikasi android dengan penambahan fase baru.
4.	*A Proposal Of An Ontology-Based Methodological Framework For Multi-Platform Mobile Applications Development* (STAPIĆ, 2013)	Pengembangan perangkat lunak berbasis *mobile* dengan metode Mobile-D	Pengembangan aplikasi *mobile* Android dan Windows phone dengan metode Mobile-D	Melakukan adopsi metode Mobile-D untuk mengembangkan aplikasi Android yang kemudian hasil dari artifacts pengembangan aplikasi android di *reuse* pada pengembangan aplikasi windows phone.

Kajian penelitian yang pertama, dilakukan oleh (Corral, et al., 2013). Coral mengatakan bahwa berdasarkan hasil penelitiannya, peneliti setuju untuk mengidentifikasi Agile sebagai metode yang paling sesuai untuk melakukan proyek perangkat lunak dalam konteks pengembangan *mobile*. Pada tingkat abstraksi yang disajikan dalam metodologi pengembangan, metode Agile membawa nilai bantu yang mana mengikuti tren bisnis seluler dan pemasaran, sementara kendala spesifik mobile lainnya seperti keterbatasan sumber daya lebih mungkin dikelola melalui pedoman pengembangan. Menurut penulis dalam tulisannya, perlunya menekankan adaptasi proses dan praktik terhadap kebutuhan perangkat lunak *mobile* yang berkembang, berdasarkan asumsi yang harus ditinjau ulang karena adanya hal – hal baru. Oleh karena itu perlunya pengembangan metode Agile untuk memperbarui praktik, asumsi, dan rekomendasi untuk realitas domain *mobile* saat ini. Evolusi *platform mobile* dari alat komunikasi sederhana menjadi peralatan *end-user computing* membuat perlunya peneliti dan praktisi untuk memahami konteks dari domain *mobile* untuk menciptakan strategi terbaik untuk mengembangkan perangkat lunak *mobile*.

Kajian penelitian yang kedua, dilakukan oleh (Abrahamsson, et al., 2004). Abrahamsson mengusulkan metode *Mobile-D* yang merupakan suatu metode pengembangan khusus aplikasi *mobile* dengan menggunakan pendekatan metode *Agile*. Metode ini didasasarkan pada beberapa metode pengembangan yaitu

Extreme Programming (XP), Metodology Crystal, dan *Rational Unified Process (RUP). Mobile-D* terdiri dari lima fase, yaitu: *Explore, Initialize, Productionize, Stabilize,* dan *System Test & Fix*. Metode pengembangan *Mobile-D* telah diuji secara empiris dan dikembangkan di laboratorium di *VTT, The Technical Research Centre* Finlandia. Didalam penelitiannya yang menerapkan pendekatan *Mobile-D*, Abraham mendapat beberapa hasil yang positif yaitu peningkatan kemajuan visibilitas, identifikasi awal dan pemecahan masalah teknis, tanggung jawab bersama, berbagi informasi yang efisien, koherensi proses yang tingggi, kesalahan produk rendah dan ritme pengembangan yang konstan.

Kajian penelitian yang ketiga, dilakukan oleh (Sapataru, 2010). Sapataru mencoba mengadopsi praktik dari metode Mobile-D tersebut. Adopsi dari metode tersebut digunakan untuk pengembangan perangkat lunak *mobile* dengan penambahan fase di akhir alur fase Mobile-D. Fase yang ditambahkan adalah *Evolve*, dimana fase tersebut dimaksudkan untuk terus terintegrasi oleh pengguna akhir aplikasi setelah aplikasi rilis. Fase tersebut juga diharapkan agar pengguna selalu memberikan *feedback* terhadap produk sebagai dasar analisis data untuk pertimbangan rilis pembaruan produk kedepannya. Pada kajian ini, Sapataru mencoba memperdalam metode Mobile-D untuk mencakup proses tidak sebatas saat pengembangan, namun juga setelah rilis produk.

Kajian penelitian yang keempat, dilakukan oleh (STAPIĆ, 2013). Stapic melakukan adopsi terhadap metode Mobile-D namun tanpa *task* yang berhubungan dengan manajemen tim. Penelitian ini berfokus pada penggalian kebutuhan yang detail untuk mendapatkan analisis kebutuhan yang matang untuk membangun aplikasi. Pada penelitiannya Stapic mengadopsi Mobile-d untuk membuat dua aplikasi dari *platform* berbeda namun dengan analisis kebutuhan yang sama dari penelitian pertama. Kemudian hasil dari pengembangan aplikasi tersebut dibandingkan untuk mengetahui waktu pengembangan aplikasi. Kesimpulan yang dibuat oleh Stapic adalah pengembangan pada aplikasi kedua menggunakan waktu lebih singkat walaupun di implementasikan pada *platform* yang berbeda, dikarenakan kebutuhan sistem tidak perlu di definisikan ulang dari awal hanya kebutuhan minor dan arsitektur perangkat *mobile* saja yang di definisikan dari awal.

2.2 Agile Development

Agile software engineering merupakan kombinasi dari filosofi dan seperangkat pedoman pengembangan. Filosofi yang mendorong kepuasan pelanggan dan pengiriman tambahan perangkat lunak lebih awal, motivasi tim yang sangat tinggi, metode tidak formal, kerja produk perangkat lunak yang minim, dan kesederhanaan pembangunan secara keseluruhan. Pedoman pengembangan menekankan pada pengiriman lebih dari analisis dan desain (walaupun kegiatan ini tidak dianjurkan) dan aktif serta komunikasi yang berkelanjutan antara pengembang dan pelanggan (Pressman, 2010).

Menurut Ian Sommervile(2011) dalam bukunya yang berjudul Software Engineering, metode agile merupakan metode pengembangan yang

“incremental” dimana pertumbuhan tersebut kecil dan biasanya rilis baru dari sistem dibuat dan bersedia bertemu dengan pelanggan setiap dua atau tiga minggu. Mereka melibatkan pelanggan dalam proses pembangunan untuk mendapatkan umpan balik yang cepat pada perubahan Kebutuhan. Mereka meminimalkan dokumentasi dengan menggunakan komunikasi tidak resmi daripada pertemuan formal dengan dokumen tertulis.

Terdapat 4 hal yang diutamakan atau perjanjian dari pengembang dalam metode agile yaitu: Individu dan interaksi lebih dari proses dan sarana perangkat lunak, Perangkat lunak yang bekerja lebih dari sekedar dokumentasi yang menyeluruh, Kolaborasi dengan klien lebih dari negosiasi kontrak, Tanggap terhadap perubahan lebih dari mengikuti rencana.

Selain itu juga terdapat 12 prinsip dalam pengembangan perangkat lunak menggunakan pendekatan metode agile seperti yang dikutip dari situs agilemanifesto.org yaitu :

1. Prioritas utama adalah memuaskan pengguna dengan menghasilkan perangkat lunak yang bernilai secara dini dan rutin.
2. Menyambut perubahan kebutuhan, walaupun pada akhir dalam pengembangan perangkat lunak. Proses Agile memanfaatkan perubahan untuk keuntungan kompetitif klien.
3. Menghasilkan perangkat lunak yang bekerja secara rutin, dari jangka waktu beberapa minggu sampai beberapa bulan berikutnya, dengan preferensi kepada jangka waktu yang lebih pendek.
4. Rekan bisnis dan pengembang perangkat lunak harus bekerja-sama sepanjang proyek.
5. Kembangkan proyek pada individual yang termotivasi. Berikan mereka lingkungan dan dukungan yang mereka butuhkan, dan percayai mereka untuk menyelesaikan pekerjaan dengan baik.
6. Metode yang paling efisien dan efektif untuk menyampaikan informasi dari dalam tim pengembangan perangkat lunak adalah dengan komunikasi secara langsung.
7. Perangkat lunak yang bekerja adalah ukuran utama kemajuan.
8. Proses Agile menggalakkan pengembangan berkelanjutan. Sponsor-sponsor, pengembang-pengembang, dan pengguna-pengguna akan dapat mempertahankan kecepatan tetap secara berkelanjutan.
9. Perhatian yang berkesinambungan terhadap keunggulan teknis dan rancangan yang baik meningkatkan Agility.
10. Kesederhanaan--seni memaksimalkan jumlah pekerjaan yang belum dilakukan--adalah hal yang penting.
11. Arsitektur, kebutuhan, dan rancangan perangkat lunak terbaik muncul dari tim yang yang dapat mengorganisir diri sendiri.

12. Secara berkala, tim pengembang berefleksi bagaimana untuk menjadi lebih efektif, kemudian menyesuaikan dan menyelaraskan kebiasaan bekerja mereka.

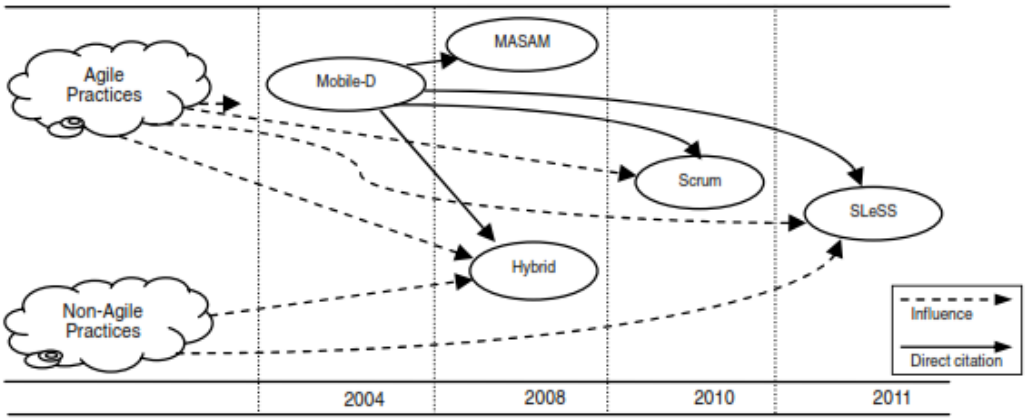
Terdapat beberapa metode agile, meskipun mungkin yang paling terkenal adalah extreme programming. Metode agile lainnya adalah Scrum, Crystal, Adaptive Software Development, DSDM, dan Feature Driven Development. Keberhasilan metode ini telah menyebabkan untuk integrasi dengan beberapa metode pengembangan yang lebih tradisional berbasiskan model sistem, yang mengakibatkatkan gagasan agile modeling dan instansiasi agile dari Rational Unified Process (Sommerville, 2011).

2.2.1 Agile Development for *Mobile*

Industri telekomunikasi *mobile* terdiri dari lingkungan yang sangat kompetitif, dinamis dan tidak pasti. Pendekatan *agile* dipandang cocok sebagai pengembangan aplikasi *mobile* yang alami. Menurut Holler(2011) yang dikutip (Flora & Chande, 2013), studi dilakukan untuk penerapan pendekatan pengembangan *agile* untuk pengembangan aplikasi *mobile* menunjukkan kebutuhan untuk proses pengembangan perangkat lunak yang disesuaikan sesuai kebutuhan aplikasi *mobile*. Telah direkomendasikan bahwa praktek *agile* adalah pilihan terbaik yang menjamin fase yang berbeda dari siklus pengembangan perangkat lunak dan untuk memecahkan masalah pengembangan aplikasi *mobile* yang lebih efisien (Flora & Chande, 2013). Ia juga meyakini bahwa inovasi metode *agile* dapat menawarkan berbagai solusi untuk aplikasi *mobile* dan membantu pengembang yang membutuhkan proses pengembangan dengan kualitas tinggi.

Abrahamsson menunjukan ciri – ciri alasan mengapa teknologi *agile* yang cocok dalam pengembangan perangkat lunak *mobile*. Berbagai isu termasuk, tingginya volatilitas lingkungan, tim pengembangan yang kecil, pelanggan yang dapat diidentifikasi, lingkungan pengembangan berorientasi objek, perangkat lunak penting non-keamanan, perangkat lunak tingkat aplikasi, sistem kecil dan siklus pengembangan singkat. Kannan juga menyoroti pengembangan perangkat lunak *agile* dalam pengembangan aplikasi *mobile* karena tim yang kecil, tenggat waktu yang singkat, pentingnya *usability*, pengiriman cepat dan kurangnya kompleksitas. Para penulis telah menyarankan tujuh metode dimana praktik pembangunan *agile* meningkatkan pengembangan aplikasi *mobile* yang mencakup eksperimen dan adaptasi alami dari aplikasi *mobile*, kehandalan yang mengarah ke penggunaan lanjutan dari aplikasi, perpanjangan sprint *Agile* ke dalam model aplikasi *mobile*, tanggap terhadap perubahan teknologi; cepat menampung umpan balik pelanggan, pengalaman pengguna yang lebih diperhatikan, dan rilis bertahap dari seperangkat fitur.

Sebagai hasilnya, ditemukan lima pendekatan agile yaitu *Mobile-D, MASAM, Hybrid Methodology, Scrum,* dan *Scrum Lean Six Sigma* yang diurutkan berdasarkan tanggal publikasi seperti pada gambar 2.1 .



Gambar 2.1 Evolusi dari Metodologi Pengembangan Agile untuk Perangkat Lunak *Mobile*

Sumber : (Corral, et al., 2013)

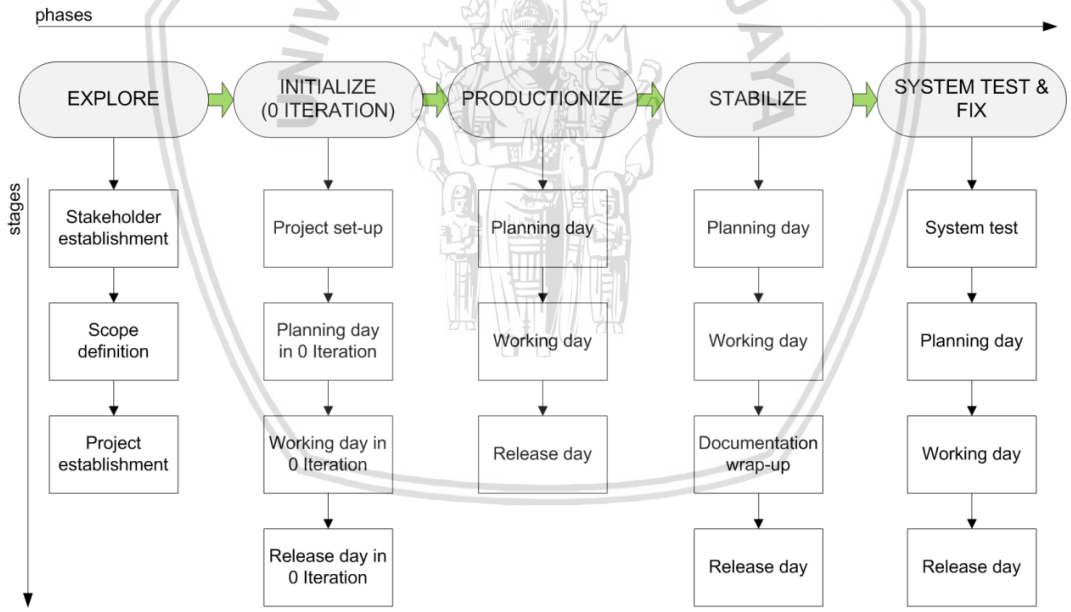
2.3 *Mobile*-D

Mobile-D merupakan upaya pertama dari penggabungan metode *agile* untuk pengembangan aplikasi *mobile*. *Mobile*-D pertama dikenalkan oleh Abrahamsson et al pada tahun 2004 sebagai metodologi pengembangan yang terinspirasi dari *Extreme Programming* (praktik pengembangan), *Crystal Methodologies* (skalabilitas) dan *Rational Unified Process* (jangkauan siklus hidup). Terdapat sembilan prinsip dalam pengembangan aplikasi dengan metode *Mobile*-D, seperti yang dikutip dari (VTT Electronics, 2004) sebagai situs pengembangan *agile Mobile*-D, yaitu:

1. *Phasing & Pacing*
   Projek dilakukan/dilaksanakan pada iterasi yang masing – masing dimulai dengan perencanaan (*planning day*)
2. *Architecture Line*
   Menggunakan pendekatan garis arsitektur (*Architecture line*) bersama – sama dengan pola arsitektural dan agile modeling.
3. *Mobile Test Driven Development*
   Pendekatan pengujian di awal (*test-first*) digunakan bersama – sama dengan kasus uji otomatis
4. *Continuous Integration*
   Menerapkan praktek SCM yang efektif melalui beberapa cara
5. *Pair Programming*
   Coding, pengujian dan refactoring dilakukan berpasangan
6. *Metrics*
   Beberapa metrik penting dikumpulkan dengan seksama dan dimanfaatkan untuk tujuan umpan balik (*feedback*) dan proses perbaikan.

7. *Agile Software Process Improvement*
   Menggunakan lokakarya(pengenalan) pasca iterasi untuk terus meningkatkan proses pengembangan
8. *Off-site Customer*
   Pelanggan(*user*) berpartisipasi dalam Perencanaan(*planning*) dan hari rilis (*release day*)
9. *User-Centered Focus*
   Penekanan diutamakan pada mengidentifikasi dan memenuhi kebutuhan pengguna akhir

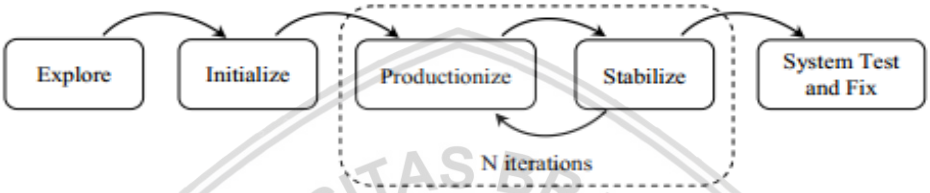
Secara singkat metodologi *Mobile-D* terdiri dari lima fase, yang masing-masing memiliki sejumlah tahapan terkait, tugas dan praktek yaitu : eksplorasi (*Explore*), inisialisasi (*Initialize*), produksi (*Productionize*), stabilisasi (*Stabilize*), pengujian dan perbaikan sistem (*System Test and Fix*). *Mobile-D* telah diterapkan dalam proyek-proyek pembangunan, dan beberapa keuntungan telah diamati, seperti peningkatan visibilitas kemajuan, deteksi lebih awal dan perbaikan masalah teknis, kerapatan cacat rendah dalam produk akhir, dan kemajuan konstan dalam pembangunan (Sapataru, 2010). Fase tersebut digambarkan dalam bagan pada gambar 2.2.



Gambar 2.2 Fase dan Tahap *Mobile*-D
Sumber : (Sapataru, 2010)

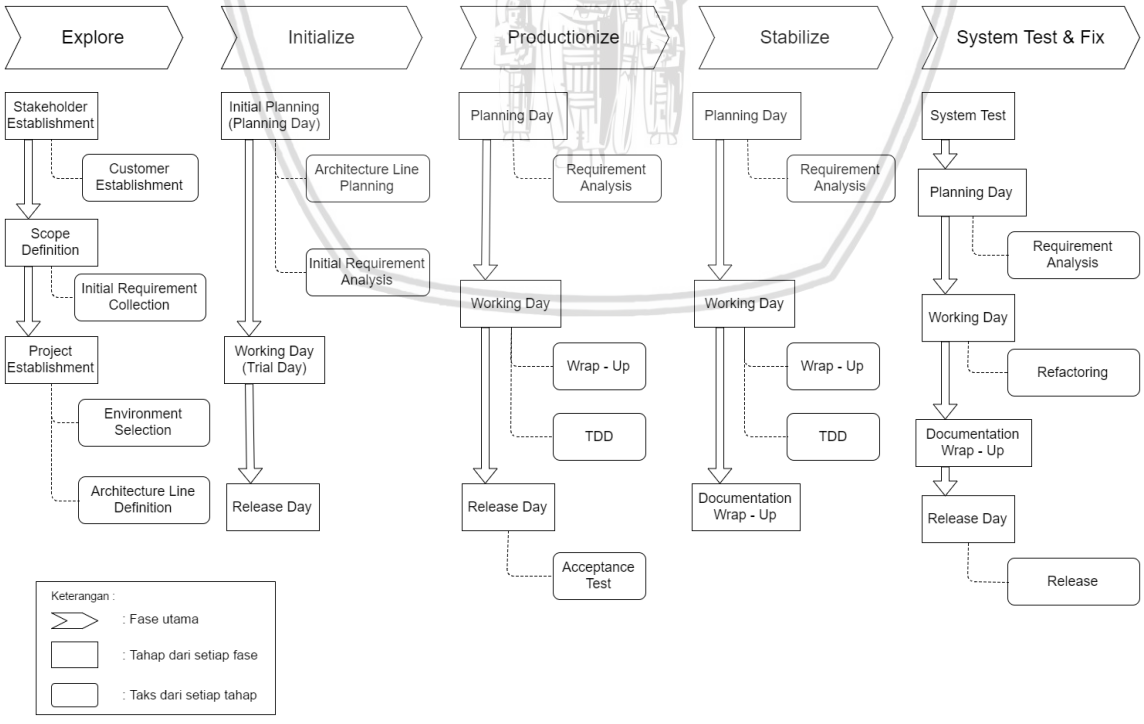
Proses pada *Mobile*-D mencakup lima tahap yang dijalankan secara parsial. Tujuan dari fase pertama, yang disebut *Explore*, adalah mempersiapkan fondasi untuk pengembangan perangkat lunak seperti identifikasi aktor yang berperan dalam pengembangan perangkat lunak. Fase kedua yaitu *Initialize*, dimana pada fase ini harus mendeskripsikan dan mempersiapkan semua komponen aplikasi serta untuk memprediksi kemungkinan isu kritis proyek. Kemudian fase *Productionize* dan *Stabilize* dijalankan secara iteratif untuk mengembangkan

semua fitur produk atau disebut juga tahap implementasi. Pada tahap ini Iterasi mulai dilakukan, yang dimulai dengan tahap *planning day* pada fase *Productionize*. Fase *Stabilize* memiliki tujuan untuk menyelesaikan implementasi, termasuk mengintegrasikan subsistem jika diperlukan. Pada fase ini bisa berisi pemrograman dan pengembangan tambahan, kegiatannya juga sangat mirip dengan aktivitas di fase *Productionize*. Terakhir, fase *System Test & Fix* bertujuan untuk mendeteksi apakah sistem yang dihasilkan mengimplementasikan fungsionalitas yang ditentukan pelanggan dengan benar. Pada fase ini juga memberikan umpan balik tim proyek mengenai fungsionalitas sistem dan informasi cacat untuk iterasi pemasangan terakhir dari proses *Mobile*-D. Proses tersebut digambarkan pada gambar 2.3 berikut :



Gambar 2.3 Proses pada *Mobile*-D
Sumber : (STAPIĆ, 2013)

Pada penelitian ini tidak menerapkan semua tahap dan *task* yang ada pada setiap fase di *Mobile*-d. Penulis hanya mengadopsi beberapa tahap dan *task* dari fase pada metode *Mobile*-d sesuai dengan kebutuhan penelitian penulis. Adopsi dari metode *Mobile*-d yang penulis gunakan digambarkan pada gambar 2.4.



Gambar 2.4 Fase, Tahap dan *Task* dari adopsi *Mobile*-d

2.3.1 Explore

Pada fase pertama *explore*, pengembang harus membuat rencana dan membangun karakteristik proyek. Pada fase ini terdiri dari tiga *stages* yang dimana setiap *stages* memiliki *task*. Tugas(*task*) terkait dengan tahap ini meliputi pembentukan pelanggan (para pelanggan yang mengambil bagian aktif dalam proses pembangunan), perencanaan awal proyek dan pengumpulan kebutuhan, dan pembentukan proses. Tujuan dari fase *explore* adalah pada perencanaan(*planning*) dan memulai pembentukan(*establish*) proyek baru. "Suatu hal yang direncanakan dengan baik sama dengan telah merampungkan setangah pekerjaan" merupakan pepatah yang untuk diingat juga dalam konteks pengembangan perangkat lunak.

Menurut (Sapataru, 2010), lokasi optimal dari tugas kategorisasi dalam metode *Mobile-D* adalah di fase *explore*, yang berisi tahap definisi cakupan(*scope definition*). Fase *explore* tidak terikat tepat waktu ke fase terakhir dari *Mobile-D* dan juga tumpang tindih dengan fase Iterasi 0(*Initialize*). Fase *explore* merupakan fase penting untuk menyusun pondasi pada pengembangan produk perangkat lunak, misalnya, masalah yang berkaitan dengan arsitektur produk, proses pengembangan perangkat lunak dan pemilihan *environment*. Kelompok pemangku kepentingan yang berbeda dibutuhkan untuk memberikan pemahaman mereka dalam fase *explore* (VTT Electronics, 2004). Fase *explore* ini terdiri dari tiga *stages* yaitu *stakeholder establishment, scope definition* dan *project establishment.*

2.3.1.1 Stakeholder Establishment

Stakeholder Establishment adalah tahap di mana semua kelompok pemangku kepentingan yang relevan – tidak termasuk tim proyek itu sendiri - diperlukan dalam pembentukan serta dalam proses pendefinisian peran dan sumber daya yang relevan. Selain kelompok pelanggan (yang telah didefinisikan dalam tugas *Customer Establishment*), *Stakeholder* di *Mobile-D* juga dapat berupa, misalnya, kelompok pengarah, manajemen proyek, kelompok arsitektur, dan spesialis proses. Semua pemangku kepentingan ini memainkan peran penting dalam tugas-tugas dari fase *explore* dan dalam pelaksanaan proyek.

Tujuan dari tahap ini adalah untuk mengidentifikasi dan membentuk kelompok-kelompok *stakeholder* yang diperlukan dalam berbagai tugas dari fase *explore* dan juga dalam mendukung kegiatan selama pengembangan perangkat lunak. Berbagai keahlian dan kerjasama diperlukan untuk merencanakan penerapan produk perangkat lunak yang terkendali dan efektif (VTT Electronics, 2004). *Task* pada tahap ini terbagi menjadi 2, yaitu *Customer Establishment* dan *Stakeholder Group Establishment*. Pada tulisan ini penulis mengadopsi *task Customer Establishment*.

1. *Customer Establishment*

Tujuan dari *task* berikut adalah untuk menentukan dasar dari komponen pelaku pengembangan perangkat lunak, mulai dari identifikasi partisipatif

pengguna, memastikan komitmen pengguna yang teridentifikasi untuk berpartisipasi dalam proyek ini, dan menentukan mode (*off/on-site*), tugas, peraturan dan tanggung jawab untuk kelompok pengguna. *Task* ini diadopsi pada penulisan bab 4 bagian identifikasi aktor.

2.3.1.2 Scope Definition

Scope Definition adalah tahap di mana tujuan dan ruang lingkup dari proyek pengembangan perangkat lunak didefinisikan dan disepakati oleh kelompok *stakeholder*. Ini termasuk isu-isu (awal) seperti kebutuhan untuk produk dan *timeline* proyek. Menurut spesifikasi *Mobile-D* pada (VTT Electronics, 2004), tahap *scope definition* mendefinisikan tujuan dan menetapkan jadwal untuk proyek. Jika tim mengidentifikasi kategori aplikasi yang mereka kembangkan, mereka dapat menetapkan tujuan proyek yang mematuhi pedoman khusus, dan dapat membentuk jadwal awal menurut data yang dikumpulkan dari proyek serupa sebelumnya. Tahap *scope definition* terdiri dari dua tugas dasar, *Initial project planning* dan *Initial requirements collection*, yang mana pada tulisan ini penulis mengadopsi *task Initial requirements collection.*

1. Initial Requirement Collection

Merupakan *task* dimana persyaratan produk harus ditetapkan pada tingkat yang sesuai. Ini harus mencakup aspek fungsional (mis., Perangkat keras dan perangkat lunak) dan juga fungsi non-fungsional (misalnya, persyaratan kualitas seperti masalah variabilitas dan pemeliharaan) persyaratan sistem. Terkadang, dimungkinkan melakukan perbaikan pada saat itu juga, namun seringkali, persyaratan baru muncul dan persyaratan awal berevolusi secara iteratif selama proyek berlangsung. Sebenarnya, ini adalah default dasar pengembangan perangkat lunak Agile. Dalam situasi seperti itu, beberapa kebutuhan awal harus ditetapkan untuk menentukan fungsi utama produk. Ini akan menjadi dasar untuk garis arsitektur dan juga iterasi 0 dari proses *Mobile*-D.

Tujuan dari *task* ini adalah mendefinisikan kebutuhan awal dari sistem, seperti ukuran, masalah teknis, arsitektur, dll. Juga kebutuhan awal sistem didefinisikan disini, kebutuhan fungsional maupun kebutuhan non-fungsional. Selain itu, dokumentasi kebutuhan akan menjadi titik awal bagi tim proyek untuk membangun keseluruhan tampilan produk yang ada. Pelanggan dan kelompok pengarah harus menyetujui dan mendokumentasikan fungsionalitas utama produk sebagaimana terlihat dari sudut pandang pelanggan. Selain itu, kebutuhan lainnya, seperti kebutuhan bisnis organisasi sendiri, dan batasan pada pengembangan produk harus diidentifikasi, disepakati dan didokumentasikan. Adopsi dari *task* ini adalah penulisan bab 4, dimana penulis melakukan penggalian kebutuhan sistem dan juga menjabarkan spesifikasi kebutuhan, membuat diagram *Use Case* maupun skenario *Use Case*.

2.3.1.3 Project Establishment

Project Establishment adalah tahap untuk menyepakati isu-isu lingkungan dari proyek (fisik dan teknis) serta personil yang diperlukan dalam pengembangan

perangkat lunak (pengembang dan dukungan). Masalah proses juga didefinisikan pada tahap ini. Tujuan dari tahap ini adalah untuk menentukan dan mengalokasikan sumber daya (baik teknis maupun manusia) yang dibutuhkan untuk pengembangan proyek perangkat lunak baru. Juga pembentukan proses dasar untuk proyek merupakan tugas penting dari tahap ini. Tahap ini terdiri dari 4 *task*, yaitu *Environment Selection, Personel Allocation, Architecture Line Definition* dan *Process Establishment*. Pada tulisan ini penulis mengadopsi *task Environment Selection*.

1. *Environment Selection*

Merupakan tugas di mana proyek direncanakan mengenai lingkungan teknis (termasuk, misalnya, perangkat target, alat pengembangan dan *platform* untuk produk perangkat lunak seluler) serta sumber daya manusia dan lingkungan kerja yang dibutuhkan di dalam proyek. *Task* tersebut diadopsi pada bab 4.

2. Architecture Line Definition

Pola ini membantu menentukan garis arsitektur untuk proyek *Mobile*-D. Masalah arsitektural dan potensi sumber risiko arsitektural yang harus dieksplorasi sebelum produksi aplikasi sebenarnya adalah, misalnya, konteks sistem, *platform* perangkat lunak, kemungkinan abstraksi inti untuk sistem, driver arsitektur yang membentuk arsitektur sistem yang sedang dibangun. , kualitas produk internal, integrasi perancangan perangkat lunak dengan proses pengembangan, mendokumentasikan arsitektur dan desain perangkat lunak dalam dokumen proyek akhir, dan keterampilan arsitektural yang dibutuhkan. *Task* ini diadopsi pada bab 4 gambaran umum sistem.

2.3.2 Initialize

Pada fase *Initialize*, pengembang dan semua pemangku kepentingan aktif memahami produk dalam pengembangan dan mempersiapkan kunci dari sumber kebutuhan untuk aktivitas produksi, baik secara fisik, teknologi dan sumber komunikasi. Tujuan dari pola fase *Initialize* adalah untuk memungkinkan keberhasilan proyek fase yang akan datang dengan mempersiapkan dan memverifikasi semua masalah pengembangan yang penting sehingga mereka semua berada dalam kesiapan penuh di akhir fase untuk menerapkan kebutuhan yang dipilih oleh pelanggan (VTT Electronics, 2004).

Fase *Initialize* ini biasanya juga dikenal sebagai iterasi 0. Gagasan fase 0 iterasi adalah memastikan fungsionalitas lingkungan pengembangan teknis melalui penerapan beberapa fitur representatif. Selain itu, pada tahap ini beberapa prototipe dapat dilakukan untuk menentukan solusi teknologi mana yang paling sesuai untuk sisa proses pembangunan (STAPIĆ, 2013).

Fase ini dibagi menjadi 4 tahap, selain *project set-up* fase ini juga terdiri dari *planning day, working day* dan *release day* yang juga terdapat pada fase *Productionize* dan *Stabilize*. Pada fase ini penulis hanya mengadopsi tahap *planning day(initial planning), working day(trial)*, dan *release day*.

2.3.2.1 Initial Planning (Planning Day)

Tujuan dari tahap *Initial Planning* adalah untuk :

1. Mendapatkan pemahaman secara keseluruhan baik dari produk untuk tim proyek berdasarkan kebutuhan awal dan deskripsi garis arsitektur(*architecture line*)
2. Memperbaiki dan memperjelas deskripsi garis arsitektur dan rencana proyek
3. Membuat rencana garis arsitektur
4. Menyiapkan rencana untuk memeriksa keadaan kesiapan masalah pengembangan yang penting seperti lingkungan pengembangan, elemen arsitektur utama dan entitas eksternal lainnya dari software yang akan dikembangkan serta komunikasi antara unsur-unsur dan entitas.

Pada tahap ini terdiri dari *task Architecture Line Planing* dan *Initial Requirement Analysis*.

1. Architecture Line Planing

Tujuan *Architecture Line Planing* adalah untuk mempersiapkan semua arsitektur perangkat lunak yang penting sehingga dalam mengembangkan sistem telah memiliki kesiapan penuh untuk pertumbuhan arsitektural yang sistematis saat menerapkan kebutuhan yang dipilih oleh pelanggan selama fase proyek yang akan datang. Adopsi dari *task* ini adalah penulisan perancangan arsitektur sistem pada bab 5.

2. Initial Requirement Analysis

Task ini berguna untuk secara hati-hati memprioritaskan dan menganalisis kebutuhan untuk menemukan seperangkat kebutuhan yang diperhitugkan untuk menciptakan komponen dan antarmuka sistem yang paling penting. Kerangka arsitektur yang bekerja harus ditemukan selambat-lambatnya pada akhir iterasi pertama. Tujuan dari *task* ini adalah memastikan bahwa kebutuhan tersebut memberikan nilai bisnis yang penting. Hasil dari *task Initial Requirement Analysis* mencakup *backlog* produk, sketsa antarmuka pengguna dan pengujian *acceptance* yang dihasilkan untuk setiap kebutuhan yang disajikan pada bab berikutnya. Product backlog menggambarkan fitur-fitur aplikasi yang disajikan melalui *user storie*. Setiap fitur memiliki tingkat kepentingan yang telah ditetapkan. Mereka diukur dengan nilai mulai dari 1 yang berarti tidak penting hingga 5 berarti sangat penting. Contoh dari produk *backlog* di tunjukan pada gambar 2.5.

Adopsi dari *task* ini adalah pada elisitasi kebutuhan, dimana penulis melakukan wawancara kepada pengguna berdasarkan elisitasi kebutuhan sebelumnya untuk menentukan prioritas implementasi dan kemudian hasilnya di tuliskan dalam bentuk tabel pada tabel 4.3.

Features / stories		Importance
F1.1	When the application is started the news should be displayed. News should include any unread answers to the user's questions; news on activities in user's groups and other information important for current user.	3
F1.2	The news presented on the first application screen should be "links" to corresponding application functionality.	3
F2.1	Current user should be able to check all his questions, including those that have been answered already. Questions should be presented by title and short description. Other details about every question should be presented in new window after user clicks on it.	5
F2.2	User should be capable to add a new question. New questions should be defined in separate windows which should include all important information about the question (title, text and images). The images should be taken by the phone camera.	5
F2.3	User should be capable to delete his/her own question. The deletion should not be performed without user's explicit confirmation on deletion action.	3

Gambar 2.5 Contoh *Product Backlog*
Sumber : (STAPIĆ, 2013)

2.3.2.2 Trial Day (Working Day)

Tujuan dari tahap ini adalah untuk menguji coba dan mengatur lebih lanjut lingkungan pengembangan teknis dan untuk memastikan semuanya siap untuk implementasi pengembangan perangkat lunak. Selain itu, tujuannya adalah untuk menerapkan beberapa fungsi inti dari sistem (misalnya komunikasi client-server) atau menyelesaikan beberapa masalah pengembangan kritis tanpa menghasilkan kode kerja apa pun. Investigasi teknologi lebih lanjut juga dimungkinkan pada tahap ini. Jika pengembangan memutuskan untuk menerapkan beberapa fungsi pada saat ini, tidak perlu menjadi fungsi dengan prioritas tertinggi sebagaimana yang didefinisikan oleh pelanggan namun dapat dipilih berdasarkan kepentingan pengembangan, misalnya, struktur arsitektur produk. *Trial Day* membentuk fase awal untuk pengembangan pada hari yang sebenarnya(*Productionize* dan *Stabilize*) (VTT Electronics, 2004).

Menurut (STAPIĆ, 2013), gagasan melakukan trial day adalah menciptakan fungsionalitas yang akan mencakup (setidaknya dalam aspek dasar) sebagian besar elemen desain arsitektural dan juga untuk menciptakan basis untuk fitur lainnya. Sebagai aplikasi yang berorientasi pada pengguna, memiliki informasi tentang pengguna saat ini merupakan prasyarat dari hampir semua fitur lainnya yang menjadikan fitur ini sebagai fungsionalitas inti dari sistem. Akhirnya, tujuan pada trial day juga untuk memastikan fungsionalitas lingkungan pengembangan teknis melalui penerapan fitur tersebut.

Pada stage ini penulis melakukan penulisan perancangan sistem mulai dari perancangan basis data hingga antarmuka awal untuk diberikan kepada pengguna pada tahap *release day*.

2.3.2.3 Release Day

Pada tahap ini hasil perancangan berupa rancangan antarmuka selesai dan dilakukan pengujian *acceptance* untuk mengetahui *feedback* pengguna (STAPIĆ, 2013).

2.3.3 Productionize

Fase *Productionize*, terutama terdiri dari pelaksanaan kegiatan. Pada akhir dari fase ini, sebagian besar pelaksanaannya harus selesai. Tujuan dalam fase *Productionize* adalah untuk mengimplementasikan fungsionalitas yang diperlukan ke dalam produk dengan menerapkan siklus pengembangan berulang dan *incremental*. Target dari fase *Productionize* adalah untuk :

1. Menerapkan fungsionalitas dari prioritas pelanggan ke dalam produk.
2. Fokus pada fungsi inti utama, dengan menerapkan mereka dalam increment awal untuk memungkinkan beberapa siklus peningkatan.

Pada fase ini dibagi menjadi *planning days, working days*, dan *release days*.

2.3.3.1 Planning Day

Planning days ditujukan untuk meningkatkan proses pembangunan, memprioritaskan dan menganalisis kebutuhan, perencanaan isi iterasi, dan menciptakan *acceptance tests* yang akan dijalankan nanti di hari rilis. Bagian penting dari tahap *planning day* adalah untuk menentukan isi (yaitu *story* dan *task*) untuk iterasi. Dengan berpartisipasi aktif dalam aktivitas perencanaan, pelanggan memastikan bahwa kebutuhan yang memberikan nilai bisnis paling banyak diidentifikasi dan kebutuhan tersebut dapat dipahami dengan benar.

Pada tahap ini terdapat beberapa task seperti Post Iteration Workshop, Requirement Analysis, Iteration planning, Acceptance Test Generation dan Acceptance Test Review. Pada tulisan ini, penulis mengadopsi task Requirement Analysis.

1. *Requirement Analysis*

Tujuan Analisis kebutuhan adalah dengan hati-hati memprioritaskan dan menganalisis kebutuhan yang dipilih untuk setiap iterasi. Selama tugas ini dipastikan bahwa kebutuhan yang memberikan nilai bisnis paling banyak diidentifikasi, dan kebutuhan tersebut dipahami dengan benar. Adopsi dari *task* ini pada penelitian berikut terletak pada bab 4 dan bab 5, dimana peneliti memperbarui kebutuhan sistem berdasarkan hasil dari iterasi 0. Iterasi 0 merupakan tahap yang dilakukan pada fase *initialize task release day* untuk mengetahui penilaian pengguna terhadap spesifikasi kebutuhan dari daftar kebutuhan dan juga dari perancangan antarmuka aplikasi.

2.3.3.2 Working Day

Tujuan dari tahap ini adalah menerapkan fungsionalitas sistem yang direncanakan pada perencanaan. Tim pengembang berfokus pada fungsi dengan prioritas tertinggi sebagaimana didefinisikan oleh pelanggan selama *planning day*. Tahap ini juga bisa dikatakan sebagai tahap implementasi. Pada tahap ini terdapat beberapa *task* yaitu, *Wrap-up, test driven development, pair programming, continuous integration, refactoring* dan *inform customer*. Namun penulis hanya mengadopsi *task Wrap-up* dan *test driven development* saja pada tulisan ini.

1. *Wrap-Up*

Tujuan dari tugas ini adalah untuk menghasilkan dokumentasi. Perangkat lunak tanpa dokumentasi adalah bencana. Kode program bukanlah media ideal untuk mengkomunikasikan penjelasan, struktur dan antarmuka sebuah sistem. Dokumentasi akan diproduksi untuk pemangku kepentingan proyek dan bukan untuk tim pengembang. Adaopsi dari *task* ini adalah pada bab 4 dan bab 5 pembaruan penulisan makalah sesuai hasil iterasi 0 untuk di implementasikan pada sistem.

2. *Test Driven Development*

Di TDD tes unit ditulis sebelum kode program. Kode program ini kemudian dikembangkan untuk bekerja dengan tes yang sudah ditulis. Tujuan TDD adalah untuk memberi pengembang kepercayaan bahwa kode yang mereka hasilkan bekerja dan membimbing desain kode agar lebih jelas strukturnya lebih mudah diuji.

2.3.3.3 Release Day

Tujuan dari tahap ini adalah sepenuhnya untuk membuat rilis kerja dari sistem yang sedang dikembangkan. Biasanya *release day* berujung ke rilis yang sebenarnya, tetapi rilis mungkin tidak resmi sehingga produk dievaluasi hanya antar *stakeholder* proyek inti. Beberapa tujuan dari tahap ini adalah :

1. Memastikan bahwa semua subsistem berhasil diintegrasikan ke dalam satu sistem.
2. Pastikan perangkat lunak yang diproduksi siap untuk pengujian acceptance.
3. Jalankan tes acceptance untuk memverifikasi bahwa kebutuhan telah diimplementasikan dengan benar.
4. Buat daftar kebutuhan baru jika ada dan daftar cacat pada perangkat lunak.
5. Berikan kesempatan tim pengembang dan pelanggan untuk mendiskusikan bagaimana perangkat lunak harus bekerja
6. Pastikan tim telah melakukan semua kegiatan yang relevan sebelum melakukan rilis perangkat lunak (dengan melakukan *release audit*).
7. Memastikan dasar untuk pengembangan lebih lanjut (dengan menciptakan garis dasar)

Pada stage ini memiliki beberapa task yaitu System Integration, Pre-Release Testing, Acceptance Testing dan Release Ceremonies. Namun penulis hanya mengadopsi task Acceptance Testing saja pada tulisan ini.

1. *Acceptance Testing*

Tujuan dari tugas ini adalah untuk memverifikasi bahwa kebutuhan yang telah ditetapkan pengguna untuk perangkat lunak diimplementasikan dengan

benar. Selain itu juga untuk mengetahui cacat yang terdapat pada perangkat lunak (VTT Electronics, 2004).

Acceptance testing adalah pengujian formal dilakukan untuk menentukan apakah sistem perangkat lunak memenuhi kriteria penerimaan dan untuk memungkinkan pengguna dapat menentukan apakah akan menerima sistem (Perry, 2006). *Acceptance testing* dirancang untuk menentukan apakah perangkat lunak "cocok(fit)" bagi pengguna untuk menggunakan. Konsep cocok ini penting antara desain dan pengujian. Desain harus berusaha untuk membangun aplikasi agar cocok ke dalam proses bisnis pengguna.

Pada tulisan ini penulis menggunakan *User Acceptance Testing*. *User Acceptance Testing*(juga diketahui sebagi *Beta Testing*), dilakukan oleh pengguna akhir(*end-users*) dari perangkat lunak. Mereka bisa menjadi pelanggan sendiri atau pelanggan dari pelanggan. Penjelasan lebih mendalam dari *Acceptance Testing* di jelaskan pada sub bab 2.7.3 Pengujian *Acceptance*. Pada *task* ini juga di lakukan iterasi 1 untuk mengetahui *feedback* pengguna terhadap hasil implementasi.

2.3.4 Stabilize

Pada fase *stabilize* digunakan untuk finalisasi produk, tahap - tahap pada *stabilize* sama seperti pada fase *productionize*, namun dengan penambahan tahap *documentation wrap-up* untuk mengakomodasi dokumentasi pembangunan. Dikutip dari situs (VTT Electronics, 2004) tujuan dari pola fase *stabilize* adalah untuk memastikan kualitas dari implementasi proyek.

Hasil dari fase pola *Stabilize* adalah :

1. Finalisasi implementasi dari sebuah produk
2. Meningkatkan dan menjamin kualitas produk.
3. Finalisasi dokumentasi produk

Fase ini terdiri dari tahap *planning days, working days, documentation wrap-up* dan *release days*.

2.3.4.1 Planning Day

Tujuan dari tahap *planning day* adalah untuk menentukan isi (yaitu *storie* dan *task*) untuk iterasi. Pada tahap ini terdapat beberapa *task* seperti :

1. *Requirement Analysis*

Melakukan analisis kebutuhan dan memprioritaskan kebutuhan apa yang akan dikerjakan pada iterasi tersebut. Pada tahap *stabilize* ini analisis kebutuhan dimaksudkan pada penambahan fungsional sistem yang telah didefinisikan tadi sesuai dengan prioritasnya. Sama seperti pada fase *productionize*, adopsi dari *task* ini pada makalah terletak pada bab 4 dan bab 5, dimana penulis memperbarui kebutuhan sistem berdasarkan hasil dari iterasi 1.

2.3.4.2 Working Day

Tujuan tahap *working day* fase *Stabilize* adalah untuk menyelesaikan pelaksanaan implementasi produk sesuai fungsionalitas yang ada sebelumnya maupun hasil iterasi 1, serta meningkatkan dan menjamin kualitas produk.

2.3.4.3 Documentation wrap-up

Tujuan dari tahap *Dokumentation Wrap-Up* adalah untuk menyelesaikan dokumen perangkat lunak, desain dan dokumen UI. *Documentation Wrap-Up* hanya mencakup satu tugas saja, yaitu *task Documentation Wrap-Up*. Tujuan dari pola *Documentation Wrap-Up* adalah untuk menyelesaikan arsitektur perangkat lunak, desain dan dokumen UI, oleh karena itu *task* ini memiliki sifat yaitu :

1. Mereka pendek, penting dan berguna.
2. Dapat dimengerti, yaitu menyampaikan informasi arsitektur, desain dan UI tentang perangkat lunak yang dikembangkan secara lengkap, konsisten dan formal sehingga orang lain dapat dengan mudah memahami apa yang dibangun, mengapa dibangun dengan cara tersebut dan juga bagaimana cara mengoperasikannya.
3. Mereka koheren(saling terkait) dengan kode program.

2.3.4.4 Release Day

Tujuan tahap Release day adalah untuk memverifikasi dan memvalidasi fungsionalitas dan kualitas yang diimplementasikan dari seluruh perangkat lunak dan dokumentasinya. Release day berujung pada rilis terakhir dari keseluruhan perangkat lunak dan siap untuk dilakukan pengujian sistem secara keseluruhan.

2.3.5 System Test and Fix

Fase terakhir adalah *System Test & Fix*, dikutip dari (VTT Electronics, 2004) fase ini digunakan untuk melihat apakah sistem yang dihasilkan telah menerapkan fungsionalitas yang ditentukan pelanggan dengan benar, berikan *feedback* kepada tim pengembang mengenai fungsionalitas sistem dan perbaikan pada cacat yang ditemukan. Tujuan dari *System Test & Fix* adalah untuk :

1. Uji sistem berdasarkan dokumentasi yang dihasilkan dalam proyek
2. Berikan informasi tentang cacat yang ditemukan
3. Biarkan tim proyek merencanakan perbaikan untuk menemukan cacatnya
4. Perbaiki cacatnya
5. Menghasilkan sistem bebas kesalahan semaksimal mungkin.

Tahap pada *System Test & Fix* mirip pada fase *productionize* maupun *stabilize*, *planning day, working day, release day* dan dengan penambahan *system test* untuk mengakomodasi pengujian sistem.

2.3.5.1 System Test & Fix

Pengujian sistem(*System Test*) adalah tahap dimana sistem diuji seperti yang dijelaskan dalam pola tugas sistem Uji. Cacat yang ditemukan didokumentasikan untuk tujuan iterasi Perbaikan(*Fix iteration*). Perbaikan(Fix) adalah variasi dari iterasi normal; Namun tidak ada fungsionalitas baru yang diimplementasikan dan untuk skala waktu khususnya, mungkin lebih pendek. Masukan untuk iterasi ini adalah cacat yang ditemukan dalam tahap uji Sistem.

Tujuan dari tugas ini adalah untuk menemukan cacat pada perangkat lunak yang diproduksi setelah tahap implementasi proyek. Prosedur Uji Sistem memberikan informasi cacat untuk iterasi perbaikan terakhir dari proses *Mobile-D*.

2.3.5.2 Planning Day

Tujuan tahap *planning day* pada fase *System Test & Fix* adalah untuk menentukan isi untuk iterasi perbaikan(*Fix*). Cacat yang ditemukan di tahap tes sistem adalah masukan untuk deskripsi tugas. Daftar cacat sistem tersebut di tuliskan pada *task requirement analysis*.

2.3.5.3 Working Day

Tujuan dari tahap ini adalah memperbaiki cacat yang ditemukan di tahap Uji Sistem dan untuk menyelesaikan implementasi produk. Pada tahap ini penulis mengadopsi *task refactoring*.

1. *Refactoring*

   Refactoring adalah proses memperbaiki struktur internal perangkat lunak yang ada tanpa memodifikasi perilaku eksternalnya. Dengan sedikit perbaikan kode, *refactoring* memastikan perangkat lunak lebih dapat dimodifikasi, dapat diperpanjang, dan mudah dibaca.

2.3.5.4 Documentation Wrap up

Tujuan tahap *Documentation Wrap up* pada fase *System Test & Fix* adalah untuk menyelesaikan dokumen perangkat lunak, desain dan dokumen UI. Dokumentasi diperbarui agar sesuai dengan perubahan yang dilakukan selama iterasi, khususnya setelah iterasi perbaikan(*Fix iteration*) pada fase ini.

2.3.5.5 Release day

Release day pada tahap ini merupakan akhir dari siklus pengembangan perangkat lunak. Dimana perangkat lunak telah final dan siap untuk di rilis kepada pengguna. Baik fungsionalitas awal maupun hasil iterasi telah selesai diimplementasikan beserta hasil dokumentasi yang final.

2.4 Android

Android adalah platform perangkat lunak dan sistem operasi untuk perangkat *mobile* yang berbasis pada *kernel Linux* dan dikembangkan oleh *Google* dan kemudian oleh *Open Handset Alliance*. Kode program untuk android tersedia

dibawah lisensi perangkat lunak bebas dan *open source*. *Android* memungkin pengembangan untuk menulis kode yang dikelola dalam bahasa Java, dan melakukan kontrol perangkat melalui *Google-developed Java libraries* (Bhardwaj, et al., 2013).

Menurut (Supriyono, et al., 2014) *android* adalah salah satu *platform* sistem operasi yang digemari masyakat karena sifatnya yang open source sehingga memungkinkan pengguna untuk melakukan pengembangan. *Android* merupakan generasi baru *platform mobile* berbasis linux yang mencakup sistem operasi, *middleware*, dan aplikasi. Arsitektur Android terdiri dari bagian-bagian seperti berikut :

a) *Applications* dan *Widgets*: layer (lapisan) dimana pengguna hanya berhubungan dengan aplikasi saja.

b) *Applications Framework*: lapisan dimana para pengembang melakukan pembuatan aplikasi yang akan dijalankan di sistem operasi Android dengan komponen-komponennya meliputi views, contents provider, resource manager, notification manager, activity manager.

c) *Libraries*: lapisan dimana fitur-fitur android berada yang berada diatas kernel meliputi library C/C++ inti seperti Libc dan SSL.

d) *Android Run Time*: lapisan yang membuat aplikasi Android dapat dijalankan dimana dalam prosesnya menggunakan implementasi Linux yang terbagi menjadi dua bagian yaitu Core Libraries dan Dalvik virtual Machine.

e) *Linux Kernel*: Layer yang berisi file-file system untuk mengatur processing, memory, resource, driver, dan sistem operasi android lainnya.

Pada tulisan ini penulis berfokus pada pengembangan aplikasi (*system apps*) pada android, yang menurupakan bagian dari arsitektur android layer *Application* dan *Application Framework.*

2.5 Firebase

Merupakan basisdata *realtime* yang bersifat *cloud hosted database* (basis data dengan komputasi awan) yang disimpan dalam format data JSON dan untuk secara secara terus menerus di sinkronisasi agar selalu mendapat keterhubungan antara server dengan klien (KhedKar & Thube, 2017). *Firebase* merupakan salah satu *webservice* yang bersifat *real-time* dan basisdata *backend* yang dapat digunakan untuk platform android, iOS dan aplikasi web. *Firebase* merupakan suatu layanan *BaaS (Backend as a Service)* yang disediakan oleh *Google*. Pada *Firebase* data disimpan dengan tipe data *JSON* dan disinkronkan secara *realtime* ke setiap klien yang menjadi *subcriber* server aplikasi *firebase*.

Firebase menyediakan beberapa fitur seperti Autentifikasi, *Hosting, Messaging, Analytics, Storage, Real-time Database, Crash Reporting, App Indexing, Test Lab,* dan lain sebagainya. Namun pada penelitian ini hanya

menggunakan fungsi autentifikasi, *Real-time Database, Database Firestore, Storage*, firebase *Function* dan *firebase Test La*b saja yang masing – masing dari fitur tersebut akan dijelaskan dibawah ini :

2.5.1 Autentifikasi

Autentifikasi adalah salah fitur dari firebase untuk memberikan izin hanya kepada pengguna yang sudah diberikan izin untuk mengakses aplikasi. Firebase menyediakan fitur autentifikasi seperti melalui nomor telepon, Gmail, Github, twitter, facebook dan juga memberikan layanan bagi pengembang untuk membuat autentifikasi sesuai dengan keinginan pengembang. Pada penelitian ini digunakan fungsi autentifikasi dengan menggunakan nomor telepon.

2.5.2 Storage

Berguna sebagai media penyimpanan yang dapat berhubungan dengan basis data *realtime*. *Storage* dapat menyimpan dan menampilkan konten seperti gambar, video dan audio. Unggah dan Unduh juga dapat dilakukan pada proses *background.* Data yang disimpan hanya dapat diakses oleh pengguna yang telah diberikan hak akses pada aplikasi tersebut. Pada penelitian ini digunakan fungsi storage untuk menyimpan gambar produk juga foto profil pengguna.

2.5.3 Real-time Database

Firebase Realtime Database adalah *database* yang di-*host* pada *cloud*. Data disimpan dalam format JSON yang tersusun dalam struktur child – child yang dapat tersimpan hingga kedalaman 32 child dan disinkronkan secara *real-time* ke setiap klien yang terhubung ke server. Ketika pengembang membuat aplikasi lintas platform dengan SDK Android, iOS, dan JavaScript, semua klien akan berbagi sebuah *instance Realtime Database* dan menerima *update* data terbaru secara otomatis. Pada penelitian ini digunakan fungsi real-time database untuk menyimpan nilai seperti data pengguna, data transaksi, data obrolan dan lainnya.

2.5.4 Firebase Firestore

Firebase Cloud Firestore adalahdatabase NoSQL yang dihosting di cloud dan dapat diakses langsung melalui SDK asli oleh iOS, Android, dan aplikasi web Anda. Selain REST dan RPC API, Cloud Firestore juga tersedia di Node.js, Java, Python, dan Go SDK. *Cloud Firestore* berfungsi untuk pengembangan seluler, web, dan server di Firebase dan Google Cloud Platform. Seperti Firebase Realtime Database, Cloud Firestore membuat data tetap terhubung di aplikasi klien melalui listener realtime dan menawarkan dukungan secara offline untuk seluler dan web. Berbeda dengan realtime database, di Cloud Firestore, data di simpan dalam struktur collection, document dan field(data).

Selain itu kelebihan firestore di banding realtime database adalah kita dapat menggunakan kueri untuk mengambil masing-masing dokumen tertentu atau semua dokumen dalam koleksi yang sesuai dengan parameter kueri Anda. Kueri Anda dapat meliputi beberapa filter berantai dan menggabungkan penyaringan dan pengurutan. Kueri juga diindeks secara default, sehingga performa kueri

sebanding dengan ukuran kumpulan hasil, bukan kumpulan data. Pada penelitian ini digunakan fungsi firestore untuk menyimpan nilai dari data produk, baik data dari produk reguler maupun produk khusus.

2.5.5 Firebase Functions

Firebase Cloud Functions berfungsi untuk dapat menjalankan kode *backend* secara otomatis sebagai respons terhadap peristiwa yang dipicu oleh fitur Firebase dan permintaan HTTPS. Setelah kita menulis dan menerapkan fungsi pada *cloud functions*, *server Google* akan segera mulai mengelola fungsi tersebut, mendeteksi peristiwa, dan menjalankannya saat fungsi dipicu. Pada penelitian ini digunakan fungsi *firebase functions* untuk melakukan sinkronasi data terhadap aplikasi pihak ketiga (algolia) saat dilakukan proses kelola produk.

2.5.6 Firebase Test Lab

Firebase Test Lab berfungsi menajalankan pengujian untuk memvalidasi aplikasi Android di berbagai jenis perangkat dan konfigurasi perangkat. Pada fungsi firebase Test Lab terdapat tiga jenis metode uji, yaitu *robo test*, *instrumentation test*, dan *game loop*. *Robo test* secara otomatis mengeksplorasi aplikasi Android pada beragam perangkat untuk menemukan cacat dan melaporkan setiap kerusakan yang terjadi. Pada *Robo test* tidak diharuskan membuat kasus uji. *Instrumentation test* dilakukan dengan menjalankan alat pengujian seperti *Espresso, Robotium,* atau *UIAutomator 2.0*, dimana perlu menulis kasus uji untuk menguji aplikasi pada beragam perangkat. Sedangkan *game loop* berguna untuk menjalankan aktivitas khusus untuk menguji kustomisasi `intent`. Pada penelitian ini digunakan metode *robo test* untuk mengetahui *compability* aplikasi pada Android API 19 hingga 26.

2.6 Unified Modelling Language

Unified Modelling Language merupakan bahasa spesifikasi dasar atau bahasa pemodelan yang digunakan dalam melakukan spesifikasi, visualisasi, membangun, dan mendokumentasikan artifak dari suatu sistem perangkat lunak. UML digunakan untuk memberikan pemahaman tentang suatu sistem yang akan dikembangkan (Rambaugh, et al., 2005).

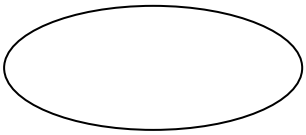
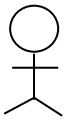
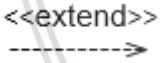
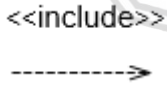
UML juga diartikan sebagai bahasa grafik standar yang bertujuan untuk memodelkan perangkat lunak berorientasi objek. James Rambaugh, Grady Booch dan Ivar Jacobson adalah yang pertama kali mengembangkan bahasa pemodelan UML pada tahun 1990, mereka mengembakan notasi sendiri. UML juga disebut sebagai bahasa pemodelan yang digunakan untuk menggambarkan suatu perancangan perangkat lunak pada pengembangan perangkat lunak yang menggunakan *Object Oriented* (OO) (Fowler, 2003).

2.6.1 *Use Case* Diagram

Use Case diagram merupakan diagram yang digunakan untuk menggambarkan sekumpulan *Use Case*, aktor yang terlibat dalam sistem serta hubungan antara

aktor dengan *Use Case*. *Use Case* merupakan sekumpulan skenario untuk mendeskripsikan bagaimana suatu sistem dapat bekerja pada sebuah situasi sesuai dengan tujuannya. Berikut ini merupakan simbol yang terdapat dalam *Use Case* diagram yang ditunjukan pada tabel 2.2

Tabel 2.2 Simbol pada *Use Case* Diagram

Simbol	Deskripsi
	***Use Case*** sekumpulan dari skenario untuk mendeskripsikan tentang cara kerja sistem pada sebuah situasi.
	**Aktor** adalah pelaku atau sistem lain yang dapat berinteraksi dengan sistem yang akan dibangun.
	Asosiasi berfungsi untuk menggambarkan interaksi antara *Use Case* dengan aktor
	**Generalisasi** digunakan ketika terdapat satu *Use Case* yang mirip dengan *Use Case* lain atau aktor yang mirip dengan aktor lain. Arah panah mengarah pada *Use Case* atau aktor yang menggeneralisasinya.
	**Extend** adalah relasi *Use Case* tambahan ke sebuah *Use Case* dimana *Use Case* yang ditambahkan dapat berdiri sendiri. Arah panah menghadap pada *Use Case* yang ditambahkan.
	**Include** adalah relasi *Use Case* tambahan ke sebuah *Use Case* dimana *Use Case* yang ditambahkan memerlukan *Use Case* atau aktor yang menjadi generalisasinya.

2.6.2 *Class* Diagram

Class diagram berguna untuk mendeskripsikan objek dan relasi diantara objek tersebut. Didalam *Class* Diagram terdapat atribut-atribut dan operasi yang ada pada suatu objek. Berikut ini merupakan simbol yang terdapat pada *class* diagram yang ditunjukan pada tabel 2.3:

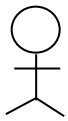
Tabel 2.3 Simbol pada *Class* Diagram

Simbol	Deskripsi
	**Nama_kelas** merupakan objek pada sistem yang memiliki atribut dan operasi. **Atribut** berguna untuk memberikan karakterisitik atau ciri yang mendeskripsikan kelas. **Operasi** adalah perilaku yang mencerminkan suatu kelas.
	Asosiasi merupakan hubungan antar kelas yang dimana suatu kelas memanggil objek dari kelas lainnya.
	Dependency merupakan hubungan antar kelas yang dimana *method* suatu kelas membutuhkan objek dari kelas lain untuk dijadikan parameter.
	Generalisasi berfungsi untuk menunjukkan hubungan *inheritance* antara anak kelas dengan induk kelas.
	**Komposisi** merupakan hubungan antar kelas yang saling bergantung satu sama lain.
	**Agregasi** adalah hubungan antar kelas yang dimana kelas tersebut merupakan bagian dari kelas lain tetapi tidak saling bergantung.

2.6.3 Sequence Diagram

Sequence diagram adalah diagram yang menggambarkan alur kontrol antara objek-objek yang terdapat dalam *Use Case*. Alur kontrol tersebut mendeskripsikan operasi-operasi yang berjalan antar objek. Berikut ini daftar simbol yang terdapat pada *sequence diagram* yang ditunjukan pada tabel 2.4:

Tabel 2.4 Simbol pada Sequence Diagram

Simbol	Deskripsi
	**Aktor** adalah pelaku atau sistem lain yang dapat berinteraksi dengan sistem yang akan dibangun.
	Pesan berfungsi untuk menyatakan suatu obyek yang memanggil operasi pada objek lain atau objek itu sendiri.

	**Boundary** juga disebut sebagai elemen antarmuka, merupakan perantara interaksi antara sistem dengan aktor.
	**Entity** merupakan klas dari domain model sistem.
	**Controller** juga dikenal sebagai elemen proses berfungsi sebagai perekat (*glue*) antara elemen boundary dan elemen entity, menerapkan logika yang diperlukan untuk mengelola berbagai elemen dan interaksinya**.**
	**Return** berguna untuk menyatakan nilai kembalian dari sebuah operasi yang dijalankan pada suatu objek.
	Lifeline adalah menyatakan kehidupan dari suatu objek.

2.7 Pengujian Perangkat Lunak

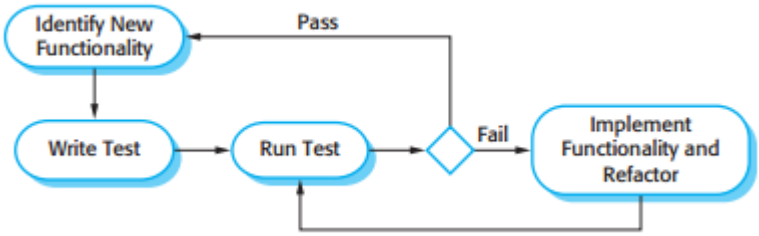
Pengujian perangkat lunak (*testing*) menghasilkan sekumpulan *layered subsystems* yang mengenkapsulasi kelas-kelas yang berkolaborasi. Setiap elemen dari sistem (subsistem dan *class*) melakukan fungsi yang bertujuan untuk mencapai kebutuhan sistem. Hal ini sangat penting untuk menguji sebuah *object-oriented system* pada berbagai macam level yang berbeda dalam sebuah usaha untuk menemukan kesalahan yang mungkin terjadi dari kolaborasi kelas-kelas dan komunikasi subsistem melewati *architetural layer*.

Perangkat lunak diuji dari dua perspektif yang berbeda: (1) logika program secara internal dilaksanakan dengan menggunakan teknik pengujian "*white box*" dan (2) kebutuhan perangkat lunak tersebut dilakukan dengan menggunakan teknik pengujian "*black box*". Use-case membantu dalam desain tes untuk mengungkap kesalahan pada tingkat validasi perangkat lunak. Dalam setiap kasus, tujuannya adalah untuk menemukan jumlah maksimum kesalahan dengan jumlah minimum usaha dan waktu (Pressman, 2010). Pada tulisan ini pengujian juga di lakukan dengan teknik *Test Driven Development* dan *Acceptance testing*.

2.7.1 Test Driven Development

Test driven development menurut (Beck, 2002; Jeffries and Melnik, 2007) yang dikutip oleh (Sommerville, 2011) adalah pendekatan pengembangan program di mana pengembang melakukan blok – blok pengujian dan pengembangan kode untuk mengujinya. Pada dasarnya, pada tdd kita mengembangkan kode secara bertahap, bersamaan dengan tes untuk kode tersebut. Kita tidak dianjurkan untuk

melanjutkan ke tahap berikutnya hingga kode yang kita kembangkan lulus uji terlebih dahulu. *Test-driven development* diperkenalkan sebagai bagian dari metode agile seperti *Extreme Programming*. Gambar 2.6 berikut menunjukan proses TDD secara dasar :



Gambar 2.6 Proses dasar TDD

Lingkungan pengujian otomatis, seperti lingkungan *JUnit* yang mendukung pengujian program *Java*, sangat penting untuk TDD. Karena kode dikembangkan bertahap dalam peningkatan sedikit demi sedikit, pengembang harus dapat menjalankan tes setiap kali selesai menambahkan fungsionalitas atau mem-refactor program. Oleh karena itu, tes dimasukkan dalam program terpisah yang menjalankan pengujian dan memanggil sistem yang sedang diuji atau fungsi yang sedang diuji. Dengan menggunakan pendekatan ini, dimungkinkan untuk menjalankan ratusan tes terpisah dalam beberapa detik. Pada penelitian ini teknik pengujian TDD digunakan untuk menguji alur logic dari perancangan algoritme, yang kemudian proses TDD tersebut dituliskan pada bab 6 sebelum pengujian unit.

2.7.2 White box

White-box testing atau *glass-box testing* merupakan sebuah metode perancangan kasus uji yang menggunakan struktur kontrol dari perancangan prosedural untuk memperoleh kasus uji. Menggunakan metode pengujian *white-box*, Anda dapat memperoleh uji kasus yang (1) menjamin bahwa semua jalur independen dalam sebuah modul telah dieksekusi setidaknya sekali, (2) melaksanakan semua keputusan logis pada sisi benar dan yang salah, (3) melaksanakan semua *loop* pada *boundaries* dan dalam batas-batas operasional mereka, dan (4) latihan struktur data internal untuk memastikan validitasnya. Ada dua jenis pengujian yang termasuk *white-box testing* yaitu *basis path testing* dan *control structure testing* (Pressman, 2010).

Basis path testing yang dikenalkan pertama kali oleh Tom McCabe. Metode *basis path* memungkinkan perancang kasus-uji untuk memperoleh ukuran kompleksitas logis dari sebuah perancangan prosedural dan menggunakan ukuran ini sebagai panduan untuk mendefinisikan basis set dari jalur eksekusi. Kasus uji diambil untuk menggunakan basis set dijamin untuk mengeksekusi setiap pernyataan dalam program setidaknya satu kali selama pengujian. *Basis path testing* terdiri dari beberapa.

2.7.2.1 Pengujian Unit

Pengujian unit (atau pengujian modul) adalah proses pengujian subprogram, subroutines, atau prosedur individual dalam suatu program. Terdapat tiga motivasi untuk melakukan pengujian ini. Pertama, pengujian unit merupakan cara mengelola elemen gabungan pengujian, karena perhatian awal difokuskan pada unit program yang lebih kecil. Kedua, pengujian unit mempermudah tugas *debugging* (proses penentuan dan mengoreksi kesalahan yang ditemukan), karena, ketika kesalahan ditemukan, maka di pastikan kesalahan tersebut ada dalam unit tersebut. Terakhir, pengujian unit memperkenalkan paralelisme ke dalam proses pengujian program dengan cara menguji beberapa unit secara bersamaan (Myers, 2004).

Pengujian unit sebagian besar berorientasi white box. Salah satu alasannya adalah bahwa ketika kita menguji entitas yang lebih besar seperti seluruh program, pengujian white-box menjadi kurang layak. Alasan kedua adalah bahwa proses pengujian berikutnya berorientasi pada proses menemukan jenis kesalahan yang berbeda (misalnya, seperti kegagalan program untuk memenuhi persyaratan penggunanya) (Myers, 2004).

2.7.2.2 Pengujian Integrasi

Pengujian integrasi adalah fase dari keseluruhan proses pengujian di mana unit perangkat lunak digabungkan dan diuji untuk mengevaluasi interaksi di antara unit tersebut. Dalam pengujian Integrasi, modul atau unit perangkat lunak terintegrasi secara logis dan diuji sebagai suatu kelompok. Pengujian integrasi berfokus pada pemeriksaan komunikasi data di antara unit-unit yang ada. Ada dua kelompok strategi integrasi perangkat lunak, pertama non incremental yang terdiri dari pendekatan big bang dan kedua incremental yang terdiri dari pendekatan Top Down Testing, Bottom Up Testing, dan sandwich (Fakultas Ilmu Komputer, 2017).

Teknik pengujian Top Down dilakukan dengan cara unit terintegrasi bergerak ke bawah melalui hiraki kontrol yang dimulai dari modul utama. *Stubs* menggantikan modul tingkat bawah di awal pengujian top-down. *Software test stubs* adalah program yang mensimulasikan perilaku komponen perangkat lunak (unit) yang tergantung berdasarkan dari unit di bawah unit uji. Sedangkan teknik pengujian Bottom Up, seperti namanya, mulai pengujian dengan unit atom (yaitu, komponen pada tingkat terendah dalam struktur program). Pada pengujian ini digunakan modul driver untuk melakukan kontrol modul. *Software test driver* adalah program yang mensimulasikan perilaku komponen perangkat lunak (unit) yang merupakan unit kontrol dari bawah unit uji (Pressman, 2010).

2.7.3 Black box

Pengujian *Black-box* atau juga dikenal *behavioral testing* berfokus pada persyaratan fungsional perangkat lunak. Pengujian *black-box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua kebutuhan fungsional untuk semua program. Pengujian *black-box* bukan merupakan alternatif dari teknik *white-box*, tetapi

merupakan pendekatan komplementer yang kemungkinan besar mampu mengungkap kelas kesalahan daripada metode *white-box*. Pengujian *black-box* berusaha menemukan kesalahan dalam kategori berikut : 1. Fungsi-fungsi yang tidak benar atau hilang 2. Kesalahan *interface* 3. Kesalahan dalam struktur data atau akses *database* eksternal 4. Kesalahan kinerja 5. Inisialisasi dan kesalahan terminasi. Tidak seperti pengujian *white-box*, yang dilakukan pada saat awal proses pengujian, pengujian black-box cenderung diaplikasikan selama tahap akhir pengujian. Karena pengujian black-box memperhatikan struktur kontrol, maka perhatian berfokus pada domain informasi. Pada penelitian ini dilakukan pengujian validasi untuk menerapkan metode black box testing.

2.7.3.1 Pengujian Validasi

Pengujian validasi adalah suatu pengujian yang bertujuan untuk mengetahui apakah sistem yang dikembangkan sudah memenuhi kebutuhan yang dibutuhkan. Pengujian validasi menggunakan teknik pengujian Black box, karena pengujian validasi lebih berfokus pada proses jalannya program serta berfokus pada input dan output apa yang di hasilkan oleh sistem.

2.7.4 Pengujian Acceptance

Pengujian *Acceptance* adalah pengujian formal dilakukan untuk menentukan apakah sistem perangkat lunak memenuhi kriteria penerimaan dan untuk memungkinkan pengguna dapat menentukan apakah akan menerima sistem (Perry, 2006). *Acceptance testing* dirancang untuk menentukan apakah perangkat lunak "cocok(fit)" bagi pengguna untuk menggunakan. Konsep cocok ini penting antara desain dan pengujian. Desain harus berusaha untuk membangun aplikasi agar cocok ke dalam proses bisnis pengguna; proses uji harus memastikan tingkat yang ditentukan sesuai(fit). Pengujian yang berkonsentrasi pada struktur dan kebutuhan memungkin gagal untuk mencapi kesesuaian(fit), dan dengan demikian gagal untuk menguji nilai dari aplikasi otomatis ke bisnis. Empat komponen dari “fit” adalah sebagai berikut:

1. *Data. Reability, timeliness, consistency* dan *usefulness* dari data termasuk ke dalam aplikasi otomatis.
2. *People*. Keterampilan, pelatihan, kecakapan, dan keinginan untuk secara layak memanfaatkan dan berinteraksi dengan aplikasi otomatis
3. *Structure*. Pengembangan yang tepat dari sistem aplikasi untuk mengoptimalkan teknologi dan memenuhi kebutuhan
4. *Rules*. Prosedur yang harus diikuti dalam pengolahan data.

Acceptance requirement dari sistem harus memenuhi dan dapat dibagi menjadi empat kategori ini :

1. *Functionality*. konsistensi internal dari dokumen dan kode dan antar tahapan; keterlacakan fungsi; verifikasi yang memadai logika; evaluasi fungsional dan pengujian; pelestarian dari fungsi di lingkungan operasi.
2. *Performance*. analisis kelayakan dari kebutuhan kinerja; simulasi dan instrumentasi alat yang tepat; analisis kinerja di lingkungan operasi.

3. *Interface quality*. Antarmuka dokumentasi; antarmuka kompleksitas; rencana uji antarmuka dan integrasi; antarmuka ergonomi; operasional pengujian antarmuka lingkungan.
4. *Overall software quality*. Kuantifikasi ukuran kualitas; kriteria penerimaan dari semua produk perangkat lunak; standar kecukupan dokumentasi dan pengembangan sistem perangkat lunak; kriteria kualitas untuk pengujian operasional.

Dikutip dari situs (Softwaretestingfundamentals, 2016), *acceptance testing* juga diketahui sebagai pengujian *black box*. *Acceptance testing* terdiri dari tiga *task*, yaitu *Acceptance test plan, Acceptance test cases/cheklist, dan Acceptance test* dimana proses pengujian dilakukan. *Acceptance testing* dilakukan setelah pengujian sistem dan sebelum membuat sistem tersedia untuk digunakan secara sesungguhnya. Terdapat dua pelaku dalam *acceptance testing* :

1. *Internal Acceptance testing* (yang juga dikenal sebagai *Alpha testing*), dilakukan oleh anggota dari organisasi yang melakukan pengembangan perangkat lunak, tapi yang tidak terlibat secara langsung kedalam proyek (pengembangan atau pengujian). Biasanya dilakukan oleh anggota manajemen produk, penjual, atau layanan pelanggan.
2. *External Acceptance testing*, dilakukan oleh orang yang berasal dari luar organisasi yang mengembangkan perangkat lunak tersebut, yang terdiri dari dua tipe, yaitu :
    a. *Customer Acceptance testing*, dilakukan oleh pelanggan dari organisasi pengembang perangkat lunak. Mereka adalah orang-orang yang meminta organisasi untuk mengembangkan perangkat lunak.
    b. *User Acceptance Testing*(juga diketahui sebagi *Beta Testing*), dilakukan oleh pengguna akhir(*end-users*) dari perangkat lunak. Mereka bisa menjadi pelanggan sendiri atau pelanggan dari pelanggan.

2.7.4.1 Pengujian Usability

Pengujian *usability* merupakan metode pengujian yang bertujuan mengetahui pertimbangan dari pengguna untuk menemukan kekurangan yang ada dalam sebuah produk perangkat lunak. Pada pengujian *usability*, pengguna akan diberikan suatu tugas untuk menjalankan produk untuk mengetahui apakah pengguna menemukan suatu kesalahan ketika menggunakan produk tersebut (Alluri, 2012). Pengujian *usability* ini mengacu pada efektivitas, efisiensi dan kepuasan pengguna.

Menurut Nielsen, *usability* adalah sebuah atribut kualitas yang dapat menilai tentang bagaimana kemudahan yang didapatkan oleh pengguna ketika menggunakan suatu produk. *Usability* didefinisikan melalui 5 komponen kualitas, yaitu :

1. *Learnability* : untuk mengukur kemudahan yang didapatkan pengguna ketika menggunakan suatu produk untuk pertama kali.
2. *Efficiency* : untuk mengukur secepat apa pengguna dapat melakukan tugasnya.
3. *Memorability* : untuk mengukur ingatan pengguna tentang proses yang dilakukan pada produk yang digunakan untuk mencapain tujuannya.
4. *Error* : untuk mengetahui kesalah yang di lakukan oleh pengguna, sejauh mana akibat dari kesalah tersebut, serta bagaimana pengguna dapat mengatasi error tersebut.
5. *Satisfaction* : untuk mengetahui tanggapan atau perasaan pengguna ketika menggunakan produk secara keseluruhan.

Pada penelitian ini menggunakan metode *System Usability Scale* untuk melakukan pengujian *usability*.

a. System Usability Scale

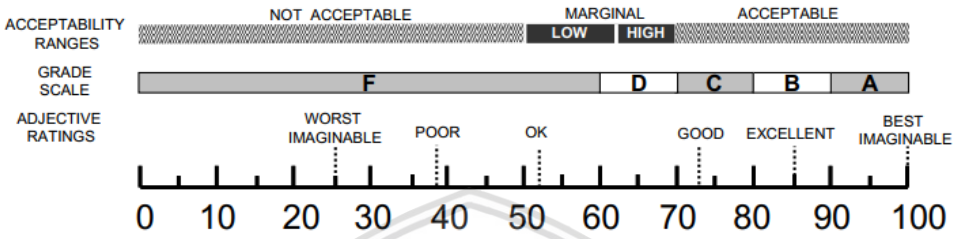
Metode selanjutnya untuk melakukan pengujian usability, salah satunya adalah System Usability Scale (SUS). SUS merupakan salah satu metode *usability* yang paling banyak digunakan saat ini (Sharfina & Santoso, 2016). SUS pertama kali dikembangkan oleh John Broke pada tahun 1986.

Penilaian yang terdapat pada SUS yaitu dengan memberikan skor untuk item bernomor ganjil seperti 1, 3, 5, 7 dan 9 dengan nilai skor kontribusi posisi skala dikurangin 1. Sedangkan untuk item bernomor genap seperti 2, 4, 6, 8 dan 10 skor kontribusinya yaitu 5 dikurangi nilai skala yang didapatkan pada tiap item genap. Selanjutnya adalah melakukan penjumlahan dari setiap skor item lalu mengkasilkan hasil penjumlahan setiap skor tersebut dengan nilai 2,5 untuk menghasilkan skor SUS. Langkah yang terakhir adalah membagi total skor SUS tadi dengan banyaknya jumlah responden, sehingga akan didapatkan nilai akhir dari skor SUS. Gambar 2.7 menunjukkan *item* asli dari SUS.

No.	Original Item
1	I think that I would like to use this system.
2	I found the system unnecessarily complex.
3	I thought the system was easy to use.
4	I think that I would need the support of a technical person to be able to use this system.
5	I found the various functions in the system were well integrated.
6	I thought there was too much inconsistency in this system.
7	I would imagine that most people would learn to use this system very quickly.
8	I found the system very cumbersome to use.
9	I felt very confident using the system.
10	I needed to learn a lot of things before I could get going with this system.

Gambar 2.7 *Original Item* SUS
Sumber : (Sharfina & Santoso, 2016)

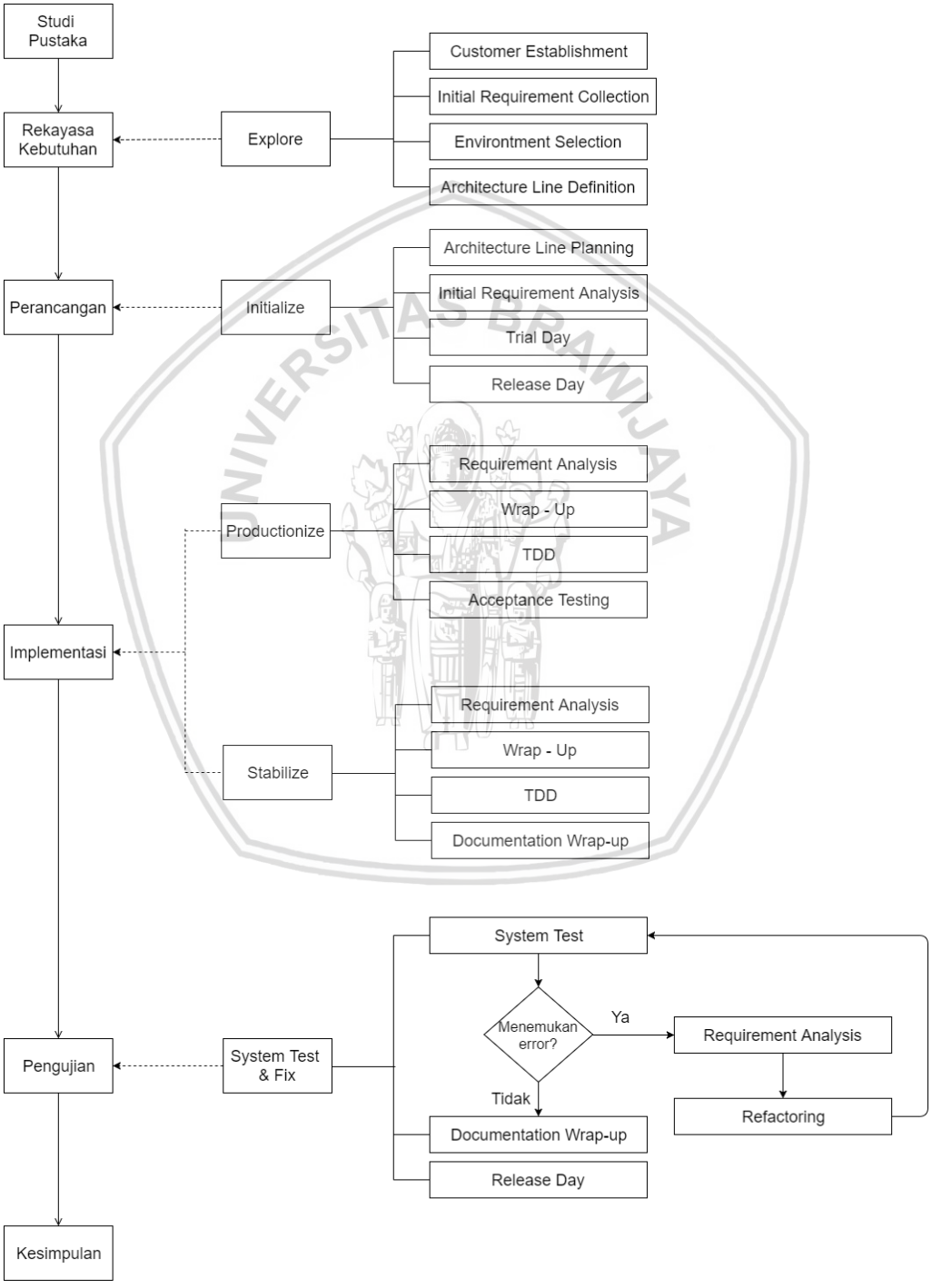
Untuk mengetahui arti dari nilai yang didapatkan dari nilai akhir SUS, terdapat range penilaian dalam SUS sebagai pengukur apakah aplikasi sudah diterima oleh pengguna atau tidak. Skor SUS dengan rentang nilai 0-50 termasuk kedalam kategori Tidak Diterima atau Not Acceptable. Skor SUS 51-70 termasuk dalam kategori Cukup atau Marginal. Selanjutnya rentang nilai skor SUS 71 – 100 yaitu termasuk ke dalam kategori Diterima atau Acceptable. Gambar 2.8 akan menjelaskan range penilaian yang terdapat pada SUS.



Gambar 2.8 Skala Penilaian SUS
Sumber : (Bangor, et al., 2009)

BAB 3 METODOLOGI

Pada bab ini dijelaskan langkah-langkah secara sistematik dalam pembangunan perangkat lunak Mlijo. Bab ini berfungsi sebagai panduan alur pengerjaan dalam penelitian yang ditujukan agar penelitian berjalan secara sistematis sesuai dengan alur tersebut. Berikut diagram alur dari langkah – langkah penelitian.



Gambar 3.1 Diagram Alur Penelitian

3.1 Studi Pustaka

Studi pustaka menjelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori - teori pendukung tersebut meliputi :

1. Kajian Pustaka
2. Agile Development
    a. *Agile Development for Mobile*
3. *Mobile*-D
    a. *Explore*
    b. *Initialize*
    c. *Productionize*
    d. *Stabilize*
    e. *System Test and Fix*
4. Android
5. Firebase
    a. *Autentifikasi*
    b. *Storage*
    c. *Real-time Database*
    d. *Firebase Firestore*
    e. *Firebase Functions*
    f. *Firebase Test Lab*
6. *Unified Modelling Language*
    a. *Use Case Diagram*
    b. *Class Diagram*
    c. *Sequence Diagram*
7. Pengujian Perangkat Lunak
    a. *Test Driven Development*
    b. *White box*
        i. *Pengujian Unit*
        ii. *Pengujian Integrasi*
    c. *Black box.*
        i. *Pengujian Validasi*
    d. *Acceptance Testing*.
        i. Pengujian *Usability*

3.2 Rekayasa Kebutuhan

Tahap rekayasa kebutuhan pada penelitian ini mengadopsi fase *explore* dari metode *Mobile-D.* Pengembang harus merumuskan dan menganalisis kebutuhan sistem untuk membangun karakteristik proyek. Pada fase ini pengembang dan semua pemangku kepentingan aktif memahami produk dalam pengembangan dan mempersiapkan segala kebutuhan untuk aktivitas produksi, mulai dari identifikasi aktor, menggali kebutuhan fungsionalitas sistem baik fungsional dan non-fungsional, pemilihan perangkat target implementasi, analisis data serta pengembang merumuskan skenario dari *Use Case*.

Pada tahap ini, adopsi dari metode *Mobile-D* terdiri dari tugas (*task) customer establishment, initial requirement collection, environtment selection* dan *architecture line definition*. Dimana pada tugas pertama mengadopsi *customer establishment*, untuk dilakukan identifikasi parsipatif pengguna dan menentukan mode elisitasi, yang pada penelitian ini menggunakan *off-site customer* atau elisitasi secara tidak langsung yang pada penelitian ini melalui survei kuisioner untuk menentukan kebutuhan sistem. Kemudian pada tugas kedua mengadopsi *initial requirement collection*, untuk mendefinisikan kebutuhan awal sistem sebagai dasar perumusan kebutuhan fungsional dan non-fungsional sistem yang kemudian akan dimodelkan dalam diagram *Use Case*. Pada tugas ketiga *environtment selection,* penulis menentukan perangkat target pengembangan perangkat lunak. Dan kemudian dilanjutkan dengan menggambarkan sistem perangkat lunak Mlijo secara umum.

3.3 Perancangan Sistem

Perancangan pada tulisan ini mengadopsi fase *Initialize* dari metode *Mobile-D*, dimana setelah fase *Initialize* selesai yang pada penelitian ini merupakan bagian perancangan sistem, maka akan terjadi iterasi 0 dengan 8 orang pengguna(5orang untuk konsumen dan 3orang untuk penjual) dari responden awal yang di pilih secara acak untuk mengetahui *feedback* dari pengguna terhadap rancangan yang telah dibuat yaitu perancangan data dan antarmuka.

Fase *Initialize* terdiri dari perancangan secara garis besar. Pada fase ini, penulis mengadopsi tahap *planning days, working days* dan *release day*. Dimana pada tahap planning days penulis mengadopsi *task architecture line planning* dan *requirement analysis*. Pada *task* pertama penulis membuat rancangan arsitektural sistem berdasarkan spesifikasi kebutuhan yang telah dibuat untuk mengetahui gambaran kebutuhan komponen penyusun aplikasi. Kemudian pada *task* kedua penulis melakukan analisis kebutuhan berdasarkan spesifikasi yang telah di rumuskan sebelumnya untuk membuat prioritas pengembangan dengan menggunakan *product backlog* 1 – 5. Dimana kebutuhan dengan nilai 5 diimplementasikan terlebih dahulu, dan kebutuhan dengan nilai di bawah 4 diabaikan/dihapus. Hasil dari analisis kebutuhan tersebut kemudian di tuliskan pada tabel 4.3

Pada tahap berikutnya penulis melakukan perancangan sistem secara keseluruhan dengan adopsi tahap *working day(trial day).* Disebut *trial day* karena pada tahap ini penulis mencoba menerapkan daftar spesifikasi kebutuhan kedalam sistem melalui perancangan basis data, diagram *class*, algoritme hingga antarmuka. Hasil dari perancangan tersebut kemudian di berikan kepada pengguna (*release day*) untuk mendapatkan penilaian pengguna. Selain itu pada tahap ini juga di mungkinkan penambahan fungsi baru untuk melengkapi fungsionalitas sistem. Hasil dari tahap ini kemudian digunakan sebagai analisis kebutuhan pada fase implementasi.

3.4 Implementasi

Implementasi pada tulisan ini mengadopsi fase *Productionize* dan *Stabilize* dari metode *Mobile-D*, dimana setelah fase *productionize* selesai yang pada penelitian ini merupakan bagian implementasi sistem, maka akan terjadi iterasi 1 dengan 5 orang pengguna(4 orang untuk konsumen dan 1 orang untuk penjual) dari responden awal yang dipilih secara acak untuk mengetahui *feedback* dari pengguna terhadap hasil implementasi awal sistem.

Pada fase *productionize* ini, penulis mengadopsi tahap *planning days, working days* dan *release day*. Dimana pada tahap *planning days* penulis mengadopsi *task requirement analysis*. Inputan pada *task requirement analysis* adalah hasil dari iterasi 0 pada bab perancangan. Pada *task* ini penulis menganalisis kebutuhan sesuai dari hasil *feedback* pengguna, dimana dimungkinkan perubahan kebutuhan seperti perubahan prioritas kebutuhan dan juga penambahan maupun penghapusan fungsionalitas sistem. Hasil dari analisis kebutuhan tersebut kemudian disajikan pada tabel 4.4. Sedangkan untuk kebutuhan dengan nilai di bawah 4 diabaikan/dihapus dari daftar kebutuhan sistem.

Tahap berikutnya adalah *wrap-up*, dimana penulis melakukan pembaruan dokumentasi sesuai dengan hasil analisis kebutuhan sebelumnya. Pembaruan dokumentasi yang dimaksud adalah seperti perbaikan diagram *Use Case*, basis data, *class*, dan lainnya sesuai dengan hasil kebutuhan terbaru. Kemudian setelah dokumentasi perancangan selesai diperbarui maka dilakukan tahap pengembangan perangkat lunak sesuai dengan urutan prioritas kebutuhan yang telah didefinisikan pada daftar kebutuhan. Sebelum melakukan implementasi kedalam kode program dilakukan perancangan algoritme kembali terlebih dahulu sebagai dasar untuk menulis kode program, yang mana pada tulisan ini mengadopsi *task Test Driven Development*(TDD). Hasil pengujian pada tahap TDD ini kemudian menjadi dasar pengujian unit dalam bentuk kode program yang hasilnya dituliskan pada bab 6.

Akhir dari fase *productionize* ini adalah tahap *release day*, dimana penulis mengadopsi *task acceptance test*. Untuk melakukan *task* tersebut, implementasi sistem secara garis besar harus telah selesai. Pada tahap ini penulis melakukan rilis awal sistem kepada pengguna untuk mengetahui *feedback* dari pengguna. Tahap ini juga di pahami sebagai tahap iterasi 1. Pada iterasi 1 ini pengguna hanya dapat menilai hasil dari implementasi dan tidak dapat melakukan perubahan

fungsionalitas mayor dari sistem. Perubahan dan penambahan fungsionalitas sistem di mungkinkan, namun hanya terbatas pada fungsi – fungsi minor saja. Hasil dari tahap ini kemudian digunakan sebagai analisis kebutuhan pada fase *stabilize*.

Pada fase *stabilize* secara garis besar sama dengan fase *productionize*, hanya terdapat penambahan tahap *documentation wrap-up* untuk menuliskan dokumentasi secara lengkap. Adopsi pada fase ini sama seperti pada fase *productionize*, hanya pada fase ini tidak melakukan tahap *release day* yang mana kemudian di gantikan oleh tahap *documentation wrap-up*. Hasil akhir dari tahap ini adalah dokumentasi lengkap dari sistem, selain itu pada tahap ini aplikasi telah final dengan telah menerapkan semua fungsionalitas sistem baik fungsionalitas awal maupun hasil iterasi. Fase berikutnya adalah pengujian sistem dan pembenahan.

3.5 Pengujian dan Analisis

Pengujian merupakan tahapan selanjutnya setelah implementasi selesai dilakukan yang juga merupakan fase terakhir dari pembangunan perangkat lunak. Pengujian sistem(*System Test*) adalah tahap dimana sistem diuji seperti yang dijelaskan dalam pola *task* sistem Uji. Cacat yang ditemukan didokumentasikan untuk tujuan iterasi Perbaikan(*Fix iteration*). Adopsi pada fase adalah tahap *system test, planning day, working day,* dan *release day*.

Pada tahap pertama *system test*, penulis melakukan pengujian secara keseluruhan, namun yang menjadi poin utama adalah bagaimana penilaian pengguna terhadap sistem tersebut baik secara fungsionalitas sistem maupun UI dari sistem. Pengujian pada tahap ini meliputi pengujian unit, pengujian integrasi, pengujian validasi, pengujian *acceptance*, dan juga pengujian *compability*.

1. Pengujian Unit merupakan pengujian yang digunakan untuk menguji setiap klas yang didapatkan dari hasil perancangan sistem. Pada penelitian ini pengujian unit dilakukan dengan menggunakan pengujuan *white box. P*engujian dilakukan dengan menggunakan *basis path testing* untuk menguji kode program berdasarkan algoritme pada setiap metode yang ada di klas.
2. Pengujian Integrasi merupakan pengujian yang digunakan untuk menguji klas-klas maupun *method* yang saling berkaitan di dalam sistem. Pada penelitian ini pengujian integrasi dilakukan dengan menggunakan pengujian *white box.* Pengujian yang dilakukan adalah dengan menggunakan *basis path testing* untuk menguji kode program berdasarkan algoritme pada setiap metode yang ada di klas.
3. Pengujian Validasi merupakan pengujian yang digunakan untuk menguji apakah seluruh kebutuhan fungsional sistem telah sesuai dengan daftar kebutuhan yang didefinisikan sebelumnya. Untuk melakukan pengujian ini menggunakan teknik pengujian *black box.*
4. Pengujian *Acceptance* merupakan salah satu metode pengujian yang dilakukan oleh pengembang terhadap pengguna dengan tujuan untuk mengetahui kepuasan pengguna terhadap sistem yang telah dikembangkan dan juga untuk

mengetahui apakah sistem dapat diterima oleh pengguna atau tidak. Pengujian *acceptance* ini menggunakan parameter *usability*. Didalam pengujian *usability* menggunakan kuisioner berdasarkan *System Usability Scale* (*SUS*) kepada pengguna. Alur pengujian di mulai dengan memperkenalkan aplikasi dan kemudian memberikan kesempatan kepada pengguna untuk menjalankan aplikasi. Pengguna kemudian diberikan 10 pertanyaan yang berkaitan dengan aplikasi seperti tampilan, alur proses , dan lain sebagainya.

5. Pengujian *compability* merupakan pengujian sistem yang bertujuan untuk memvalidasi ketergantungan antara aplikasi *mobile* yang diuji dengan lingkungan operasi yang berbeda. Pada penelitian ini dilakukan pengujian *compability native API* pada android dengan menggunakan bantuan *firebase Test Lab*. Dimana aplikasi dalam bentuk `.apk` di unggah ke sistem f*irebase Test Lab*, dan kemudian dijalankan dalam perangkat *virtual* dengan `API` android yang berbeda mulai dari `API 19` hingga `API 26`.

Kemudian setelah pengujian selesai dilakukan, jika terdapat error atau cacat pada sistem akan di lakukan perbaikan(*fix*) atau iterasi internal pengembang. Masukan untuk iterasi ini adalah cacat yang ditemukan dalam tahap uji Sistem. Daftar cacat aplikasi yang ditemukan di tuliskan pada *task requirement analysis* tahap *planning day*. Kemudian dilakukan perbaikan kode oleh internal pengembang itu sendiri yang pada tulisan ini mengadopsi *task refactoring* dari tahap *working day*. Hasil dari perbaikan kemudian di uji kembali pada tahap uji sistem. Apabila tidak ditemukan cacat sistem, maka akan dilakukan pembaruan dokumentasi sesuai dengan hasil akhir dari perangkat lunak yang pada tulisan ini mengadopsi tahap *documentation wrap-up*. Dan akhir dari fase ini adalah tahap *release day* dimana perangkat lunak secara utuh telah siap dirilis dan digunakan oleh pengguna.

3.6 Kesimpulan dan Saran

Penulisan kesimpulan dilakukan setelah seluruh tahapan pengembangan perangkat lunak terselesaikan, mulai dari tahapan perancangan perangkat lunak, implementasi perangkat lunak, dan pengujian perangkat lunak. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem yang dikembangkan. Sedangkan saran dapat digunakan sebagai sarana memperbaiki kesalahan – kesalahan yang terjadi pada penelitian serta menjadi tolak ukur pertimbangan atas penelitian selanjutnya.

BAB 4 REKAYASA KEBUTUHAN

Bab ini menjelaskan fase rekayasa kebutuhan dari pengembangan sistem katalog dan pemesanan produk kebutuhan dapur berbasis android, MLIJO. Pada tahapan pertama akan dilakukan analisis kebutuhan yang terdiri dari beberapa langkah, dimana langkah-langkah ini diadopsi dari metode pengembangan *Mobile-D* fase *explore*, yaitu melakukan proses identifikasi aktor, melakukan penggalian kebutuhan dengan melakukan survei terhadap pengguna, pemilihan perangkat pengembangan, menggambarkan gambaran umum sistem yang akan dikembangkan, analisis data dan melakukan pemodelan untuk menggambarkan kebutuhan dengan diagram *Use Case* serta melakukan analisis kebutuhan fungsional dan non-fungsional.

4.1 Analisis Kebutuhan

Proses analisis kebutuhan diawali dengan melakukan proses identifikasi aktor yang terlibat, melakukan penggalian kebutuhan, menggambarkan gambaran umum sistem yang akan dikembangkan, analisis data yang diperlukan dan melakukan analisis kebutuhan baik kebutuhan fungsional maupun non-fungsional dan kemudian digambarkan dalam diagram *Use Case* serta dijelaskan dalam skenario *Use Case*. Tujuan dari dilakukannya analisis kebutuhan adalah untuk mengetahui kebutuhan-kebutuhan yang harus disediakan oleh sistem untuk dapat memenuhi kebutuhan pengguna.

4.1.1 Identifikasi Aktor

Tahap ini merupakan tahapan untuk melakukan identifikasi aktor yang akan berinteraksi dalam sistem yang akan dibangun. Tabel 4.1 akan memperlihatkan aktor-aktor yang terlibat berserta penjelasannya.

Tabel 4.1 Identifikasi Aktor

Aktor	Deskripsi
Pengguna	Merupakan warga kota Malang pengguna sistem pemesanan kebutuhan dapur dan sayuran di Kota Malang(MLIJO) melalui *smartphone* android masing-masing yang belum terdaftar dalam sistem atau sudah terdaftar namun belum melakukan login.
Konsumen	Merupakan pengguna dari sistem pemesanan kebutuhan dapur dan sayuran di Kota Malang yang telah terdaftar dalam sistem dan telah melakukan login sehingga dapat menggunakan layanan yang disediakan dalam sistem.
Penjual	Merupakan pengguna dari sistem pemesanan kebutuhan dapur dan sayuran di Kota Malang yang telah terdaftar dalam sistem dan telah melakukan login sehingga dapat

	menggunakan layanan yang disediakan sebagai penjual produk.

4.1.2 Elisitasi Kebutuhan

Pada bagian elisitasi kebutuhan akan membahas tentang kebutuhan dari sistem katalog dan pemesanan produk kebutuhan dapur berbasis android. Proses pengumpulan kebutuhan perangkat lunak dasar didapatkan dari hasil wawancara terhadap 35 responden yang dimana responden dibagi menjadi 30 responden untuk aplikasi konsumen yang melakukan pemesanan dan 5 responden untuk aplikasi penjual sebagai pemilik atau penyedia produk. Kemudian hasil wawancara tersebut di tuliskan pada tabel definisi kebutuhan. Hasil definisi kebutuhan kemudian di perjelas lebih spesifik pada spesifikasi kebutuhan.

Setelah itu daftar spesifikasi kebutuhan di berikan lagi terhadap pengguna untuk di tentukan tingkat kepentingan (prioritas) dari kebutuhan sistem sesuai dengan penilaian pengguna.

Tabel 4.2 Definisi Kebutuhan Konsumen

No	Nama Kebutuhan
1.	Saya ingin disediakan fitur informasi harga produk terkini
2.	Saya ingin disediakan fitur Informasi terbaru lokasi penjual aktif disekitar
3.	Saya ingin disediakan fitur Informasi nilai gizi
4.	Saya ingin disediakan fitur obrolan dengan penjual
5.	Saya ingin disediakan fitur pencarian produk
6.	Saya ingin disediakan fitur halaman daftar produk
7.	Saya ingin disediakan fitur pemesanan produk
8.	Saya ingin disediakan fitur halaman daftar penjual
9.	Saya ingin disediakan fitur manajemen akun saya
10.	Saya ingin disediakan fitur untuk memberi rating dan review terhadap penjual
11.	Saya ingin disediakan fitur mengelola pembelian
12.	Saya ingin disediakan fitur Informasi produk populer
13.	Saya ingin disediakan fitur rekomendasi produk berdasarkan cuaca atau harga terkini

Tabel 4.2 diatas ini merupakan hasil dari tahap *initial Requirement Collection* dimana penulis melakukan observasi daftar kebutuhan dari pengguna untuk merumuskan definisi kebutuhan dari sistem MLIJO pada sisi konsumen. Hasil dari daftar definisi kebutuhan tersebut kemudian digunakan untuk membuat spesifikasi kebutuhan sistem baik kebutuhan fungsional dan non-fungsional.

Tabel 4.3 Elisitasi Kebutuhan Konsumen

No		Nama Kebutuhan	Prioritas Kebutuhan					Nilai Akhir
			1	2	3	4	5	
1.		Saya ingin disediakan fitur pemesanan produk	0	0	3	6	21	5
	1.1	Sistem harus dapat menyediakan fitur untuk memesan produk						
	1.2	Sistem harus dapat menyediakan fitur untuk memesan produk khusus						
2.		Saya ingin disediakan fitur halaman daftar produk	0	1	3	5	21	5
	2.1	Sistem harus dapat menampilkan daftar produk						
	2.2	Sistem harus dapat menampilkan detail produk						
3.		Saya ingin disediakan fitur informasi harga produk terkini	0	1	2	8	19	5
	3.1	Sistem harus dapat menampilkan informasi harga terkini						
4.		Saya ingin disediakan fitur Informasi terbaru lokasi penjual aktif disekitar	0	1	2	9	18	4
	4.1	Sistem harus dapat menampilkan informasi lokasi penjual aktif di sekitar area konsumen						
5.		Saya ingin disediakan fitur obrolan dengan penjual	0	1	4	6	19	4
	5.1	Sistem harus menyediakan fasilitas mengirim obrolan kepada penjual						
6.		Saya ingin disediakan fitur pencarian produk	0	1	4	6	19	4
	6.1	Sistem harus menyediakan fasilitas pencarian produk						
	6.2	Sistem harus menyediakan fasilitas filter produk						
7.		Saya ingin disediakan fitur mengelola pembelian	0	0	4	12	14	4
	7.1	Sistem harus dapat menyediakan halaman daftar transaksi						

	7.2	Sistem harus dapat menyediakan fitur detail transaksi						
	7.3	Sistem harus dapat menyediakan fitur melihat status pesanan						
	7.4	Sistem harus dapat menyediakan fitur konfirmasi penerimaan produk						
	7.5	Sistem harus dapat menyediakan fitur untuk melihat riwayat transaksi						
8.		Saya ingin disediakan fitur halaman daftar penjual	0	1	5	9	15	4
	8.1	Sistem harus dapat menampilkan daftar penjual						
	8.2	Sistem harus dapat menampilkan detail profil penjual						
9.		Saya ingin disediakan fitur untuk memberi rating dan review terhadap penjual	0	1	4	11	14	4
	9.1	Sistem harus menyediakan fasilitas untuk memberikan ulasan ke penjual						
10.		Saya ingin disediakan fitur manajemen akun saya	0	0	8	12	10	4
	10.1	Sistem harus menyediakan fitur registrasi						
	10.2	Sistem harus menyediakan fitur login						
	10.3	Sistem harus menyediakan fitur logout						
	10.4	Sistem harus meyediakan fasilitas untuk mengelola data profil						
11.		Saya ingin disediakan fitur Informasi nilai gizi	1	2	10	11	6	4
	11.1	Sistem harus dapat menyediakan fitur untuk melihat informasi nilai gizi						
12.		Saya ingin sistem dapat menyediakan fitur Informasi produk populer	3	7	12	4	4	3
	12.1	Sistem harus dapat menyediakan informasi produk populer						
13.		Saya ingin disediakan fitur rekomendasi produk berdasarkan cuaca atau harga terkini	3	8	11	5	3	3
	13.1	Sistem harus dapat menyediakan fitur rekomendasi produk kepada						

		konsumen berdasarkan kondisi cuaca atau harga produk terkini						

Tabel 4.3 diatas ini merupakan hasil dari tahap *Initial Requirement Analysis* fase *initialize* dimana penulis melakukan prioritas kebutuhan untuk membuat perancangan sistem sesuai dengan hasil kebutuhan tersebut. Kebutuhan dengan nilai kurang dari 4 tidak akan digunakan dan dihapus karena dianggap tidak penting oleh pengguna. Indikator penilaian 1 – 5 ini mengacu pada teori produk *backlog task initial requirement analysis* pada bab 2.3.2.1, dimana kebutuhan dengan nilai 1 memilik arti sangat tidak penting, sedangkan kebutuhan dengan nilai 5 memilik arti sangat penting.

Tabel 4.4 Prioritas Kebutuhan Konsumen iterasi 0

No		Nama Kebutuhan	Prioritas Kebutuhan					Nilai Akhir
			1	2	3	4	5	
1.		Saya ingin sistem dapat menyediakan fitur pemesanan produk	0	0	0	1	4	5
	1.1	Sistem harus dapat menyediakan fitur untuk memesan produk						
	1.2	Sistem harus dapat menyediakan fitur untuk memesan produk khusus						
2.		Saya ingin disediakan fitur halaman daftar produk	0	0	0	1	4	5
	2.1	Sistem harus dapat menampilkan daftar produk						
	2.2	Sistem harus dapat menampilkan detail produk						
3.		Saya ingin disediakan fitur informasi harga produk terkini	0	0	0	1	4	5
	3.1	Sistem harus dapat menampilkan informasi harga terkini						
4.		Saya ingin disediakan fitur halaman daftar penjual	0	0	0	1	4	5
	4.1	Sistem harus dapat menampilkan daftar penjual						
	4.2	Sistem harus dapat menampilkan detail profil penjual						
5.		Saya ingin disediakan fitur Informasi terbaru lokasi penjual aktif disekitar	0	0	0	2	3	5

	5.1	Sistem harus dapat menampilkan informasi lokasi penjual aktif di sekitar area konsumen						
6.		Saya ingin disediakan fitur pencarian produk	0	0	0	2	3	5
	6.1	Sistem harus menyediakan fasilitas pencarian produk						
	6.2	Sistem harus menyediakan fasilitas filter produk						
7.		Saya ingin disediakan fitur mengelola pembelian	0	0	0	3	2	4
	7.1	Sistem harus dapat menyediakan halaman daftar transaksi						
	7.2	Sistem harus dapat menyediakan fitur detail transaksi						
	7.3	Sistem harus dapat menyediakan fitur melihat status pesanan						
	7.4	Sistem harus dapat menyediakan fitur konfirmasi penerimaan produk						
	7.5	Sistem harus dapat menyediakan fitur untuk melihat riwayat transaksi						
8.		Saya ingin disediakan fitur untuk memberi rating dan review terhadap penjual	0	0	0	3	2	4
	8.1	Sistem harus menyediakan fasilitas untuk memberikan ulasan ke penjual						
9.		Saya ingin disediakan fitur manajemen akun saya	0	0	0	3	2	4
	9.1	Sistem harus menyediakan fitur registrasi						
	9.2	Sistem harus menyediakan fitur login						
	9.3	Sistem harus menyediakan fitur logout						
	9.4	Sistem harus meyediakan fasilitas untuk mengelola data profil						
10.		Saya ingin disediakan fitur obrolan dengan penjual	1	0	0	2	2	4
	10.1	Sistem harus menyediakan fasilitas mengirim obrolan kepada penjual						

11.		Saya ingin disediakan fitur Informasi nilai gizi	0	3	1	1	0	3
	11.1	Sistem harus dapat menyediakan fitur untuk melihat informasi nilai gizi						

Tabel 4.4 diatas ini merupakan hasil dari tahap *Requirement Analysis* fase *productionize* dimana penulis melakukan prioritas kebutuhan untuk menentukan fungsionalitas mana yang harus di implementasikan terlebih dahulu. Kebutuhan dengan nilai kurang dari 4 tidak akan di implementasikan dan dihapus karena dianggap tidak mendapatkan penilaian baik dari pengguna. Sama halnya dengan hasil iterasi 0 pada tabel 4.3, penilaian kebutuhan pada tabel ini juga mengacu pada teori produk *backlog* bab 2.3.2.1 *task initial requirement analysis*.

Tabel 4.5 Kebutuhan Konsumen Hasil iterasi 1

No		Nama Kebutuhan	Prioritas Kebutuhan					Nilai Akhir
			1	2	3	4	5	
1.		Saya ingin sistem dapat menyediakan fitur notifikasi	0	0	0	0	2	5
	1.1	Sistem harus dapat menyediakan fitur untuk menampilkan notifikasi						

Tabel 4.5 diatas ini merupakan hasil dari *task Requirement Analysis* pada fase *stabilize*. *Requirement Analysis* ini dilakukan setelah mendapatkan *feedback* dari pengguna pada fase *productionize*. Dimana pada tahap ini terdapat kebutuhan baru yang di tambahkan oleh pengguna. Kebutuhan tersebut kemudian di tuliskan pada daftar kebutuhan dan dibuat spesifikasi kebutuhan serta *Use Case*nya.

Tabel 4.6 Daftar Kebutuhan Penjual

No	Nama Kebutuhan
1.	Saya ingin disediakan fitur untuk mengelola produk
2.	Saya ingin disediakan fitur untuk mengelola penjualan
3.	Saya ingin disediakan fitur aktif/nonaktifkan akun pada kondisi tertentu
4.	Saya ingin disediakan fitur untuk mengelola akun saya
5.	Saya ingin disediakan informasi harga pasar terkini
6.	Saya ingin disediakan fitur chat dengan konsumen
7.	Saya ingin disediakan fitur berbagi lokasi dengan konsumen
8.	Saya ingin disediakan fitur rute perjalanan menuju lokasi konsumen

Tabel 4.6 diatas ini merupakan hasil dari tahap *Initial Requirement Collection* dimana penulis melakukan observasi daftar kebutuhan dari pengguna untuk merumuskan definisi kebutuhan dari sistem MLIJO pada sisi penjual. Hasil dari daftar definisi kebutuhan tersebut kemudian digunakan untuk membuat spesifikasi kebutuhan sistem baik kebutuhan fungsional dan non-fungsional.

Tabel 4.7 Elisitasi Kebutuhan Penjual

No		Nama Kebutuhan	Prioritas Kebutuhan					Nilai Akhir
			1	2	3	4	5	
1.		Saya ingin disediakan fitur untuk mengelola produk	0	0	0	0	5	5
	1.1	Sistem harus dapat menyediakan fitur untuk mengelola produk						
	1.2	Sistem harus dapat menyediakan fitur untuk melakukan tambah produk baru						
	1.3	Sistem harus dapat menyediakan fitur untuk merubah data produk						
	1.4	Sistem harus dapat menyediakan fitur untuk menghapus produk						
	1.5	Sistem harus dapat menyediakan fitur untuk menampilkan daftar produk						
2.		Saya ingin disediakan fitur untuk mengelola penjualan	0	0	0	0	5	5
	2.1	Sistem harus dapat menyediakan halaman daftar transaksi						
	2.2	Sistem harus dapat menyediakan fitur detail transaksi						
	2.3	Sistem harus dapat menyediakan fitur untuk melakukan konfirmasi pesanan						
	2.4	Sistem harus dapat menyediakan fitur untuk memperbarui status transaksi produk						
	2.5	Sistem harus dapat menyediakan fitur untuk melihat riwayat transaksi						
3.		Saya ingin disediakan fitur untuk mengelola akun saya	0	0	0	1	4	5
	3.1	Sistem harus menyediakan fitur registrasi						
	3.2	Sistem harus menyediakan fitur login						
	3.3	Sistem harus menyediakan fitur logout						
	3.4	Sistem harus meyediakan fasilitas untuk mengelola data profil						

	3.5	Sistem harus dapat menyediakan fitur untuk melakukan aktif/nonaktifkan akun pada kondisi tertentu						
4.		Saya ingin disediakan informasi harga pasar terkini	0	0	1	2	2	4
	4.1	Sistem harus dapat menampilkan informasi harga terkini						
5.		Saya ingin disediakan fitur kirim obrolan dengan konsumen	0	0	1	3	1	4
	5.1	Sistem harus menyediakan fasilitas mengirim obrolan kepada konsumen						
6.		Saya ingin disediakan fitur berbagi lokasi dengan konsumen	0	0	2	3	0	4
	6.1	Sistem harus menyediakan fasilitas untuk berbagi lokasi penjual kepada konsumen secara real-time						
7.		Saya ingin disediakan fitur rute perjalanan menuju lokasi konsumen	0	0	2	3	0	4
	7.1	Sistem harus menyediakan fitur untuk menampilkan peta arah menuju lokasi alamat konsumen						

Tabel 4.7 diatas ini merupakan hasil dari tahap *Initial Requirement Analysis* dimana penulis melakukan prioritas kebutuhan untuk membuat perancangan sistem sesuai dengan hasil kebutuhan tersebut. Kebutuhan dengan nilai kurang dari 4 tidak akan digunakan dan dihapus karena dianggap tidak mendapatkan penilaian baik dari pengguna. Sama seperti pada sisi konsumen, pada sisi penjual ini juga di lakukan iterasi dengan jumlah yang sama. Namun karena tidak terdapat perubahan kebutuhan maupun prioritas maka oleh penulis hanya di tuliskan satu kali saja.

4.1.3 Pemilihan Lingkungan Pengembangan

Pengembangan perangkat lunak MLIJO ini dilakukan pada lingkungan sistem operasi android. Sistem operasi android di pilih karena dari data yang di himpun dari *Statcounter* sekitar 87% pengguna perangkat *mobile* menggunakan sistem operasi *android*. Kemudian pada pengembangan dibatasi pada *Operating System Version* 4.4(kitkat) keatas atau API 19 keatas, dikarenakan sekitar 94% pengguna android di Indonesi menggunakan sistem operasi android versi tersebut sesuai dengan *Android Version Market Share* di Indonesia (Statcounter, 2017).

Pada penelitian ini, pengembangan aplikasi sisi konsumen dilakukan pada android *OS version 6.0(Marshmallow)*, hal ini didasarkan dari persentase sebaran android *OS version* yang memiliki nilai tertinggi yaitu 29.98% pengguna.

Sedangkan pengembangan aplikasi sisi penjual dilakukan pada *OS version 4.4(kitkat)* dikarenakan merupakan sistem operasi minimum.

4.1.4 Gambaran Umum Perangkat Lunak *MLIJO*

Pembahasan gambaran umum perangkat lunak *MLIJO* terdiri atas dua bagian, yaitu deskripsi umum perangkat lunak *MLIJO* dan cara penggunaan perangkat lunak *MLIJO*.

4.1.4.1 Deskripsi Perangkat Lunak MLIJO

Perangkat lunak yang akan dikembangkan dalam proyek skripsi ini adalah perangkat lunak *MLIJO*. Perangkat lunak *MLIJO* adalah perangkat lunak yang dapat digunakan untuk melihat daftar produk kebutuhan dapur yang dijual oleh penjual sayur keliling serta memesan produk tersebut. Perangkat lunak *MLIJO* diharapkan mampu mengakomodasi kebutuhan masyarakat baik konsumen dalam mencari kebutuhan dapur setiap harinya maupun penjual sayur keliling dalam berjualan.

Perangkat lunak *MLIJO* memiliki 2 bagian utama yaitu, subsitem aplikasi konsumen dan subsistem aplikasi penjual.

a. Aplikasi Konsumen

   Proses – proses utama yang terdapat dalam aplikasi konsumen dalah sebagai berikut :

   1. Melakukan pencarian produk
      Operasi pencarian ini bertujuan untuk mendapatkan data produk yang tersedia dan dijual oleh penjual.
   2. Melihat daftar produk
      Operasi ini bertujuan untuk menampilkan informasi daftar produk yang dijual oleh penjual.
   3. Melakukan pemesanan
      Operasi memesan produk ini bertujuan untuk melakukan pemesanan produk yang tersedia dalam daftar produk yang dijual oleh penjual.
   4. Mengelola pembelian
      Operasi ini bertujuan untuk mengelola data pembelian seperti melihat status pesanan dan riwayat pemesanan produk.
   5. Melihat informasi harga terkini
      Operasi melihat informasi harga terkini ini bertujuan untuk dapat mengetahui kondisi harga produk di pasaran secara baru.
   6. Melihat lokasi penjual aktif
      Operasi melihat lokasi penjual aktif ini bertujuan untuk dapat menampilkan lokasi penjual pada peta disisi konsumen sehingga konsumen dapat mengetahui lokasi penjual pada peta

7. Melihat Informasi Gizi
   Operasi ini bertujuan untuk dapat menampilkan informasi gizi sebagai edukasi bagi konsumen untuk membantu dalam memilih produk. Pada iterasi 1, operasi ini dihapus karena kurang sesuai dengan keinginan konsumen.

8. Melihat Daftar Penjual
   Operasi ini bertujuan untuk dapat menampilkan daftar dari seluruh penjual terdaftar agar konsumen dapat memilih penjual yang diinginkan.

9. Memberikan Ulasan
   Operasi membuat ulasan ini bertujuan untuk dapat memberikan penilaian terhadap penjual berupa ulasan dari konsumen sehingga dapat menjadi referensi bagi konsumen dalam memilih penyedia produk yang dipesan.

   Operasi ini terdiri dari dua operasi yaitu :

   i. Melakukan insert ulasan berupa teks
   ii. Melakukan insert ulasan berupa rating dalam bentuk *rating star*

10. Mengirimkan obrolan
    Operasi ini bertujuan untuk dapat melakukan aktivitas kirim obrolan dengan penjual.

11. Mengelola data profil konsumen
    Operasi ini bertujuan untuk dapat melakukan manipulasi data profil konsumen yang di simpan pada tabel konsumen di *database*.

    Operasi ini terdiri dari beberapa operasi, yaitu :

    i. Melakukan *insert* pada tabel konsumen
    ii. Melakukan *update* pada tabel konsumen

12. *Logout*
    Operasi ini bertujuan untuk dapat keluar dari sistem sehingga konsumen tidak dapat menggunakan fungsi – fungsi sebagai konsumen.

b. Aplikasi Penjual

Proses – proses utama yang terdapat dalam aplikasi penjual dalah sebagai berikut :

1. Mengelola data produk
   Operasi ini bertujuan untuk dapat melakukan manipulasi data produk yang disimpan pada tabel produk di *database*.
   Operasi ini terdiri dari beberapa operasi yaitu :

   i. Melakukan *insert* pada tabel produk

ii. Melakukan *update* pada tabel produk
iii. Melakukan *delete* pada tabel produk

2. Mengelola penjualan
   Operasi ini bertujuan untuk dapat melakukan manipulasi data pesanan produk dari konsumen yang di simpan pada tabel pesanan produk di *database*.

   Operasi ini terdiri dari beberapa operasi, yaitu :

   i. Melihat daftar pesanan
   ii. Melakukan *update* pada tabel pesanan produk

3. Mengelola status akun
   Operasi ini bertujuan untuk dapat mengelola akun sehingga penjual dapat merubah statusnya sedang berjualan (aktif) atau sedang libur (non-aktif).

4. Mengelola data profil penjual
   Operasi ini bertujuan untuk dapat melakukan manipulasi data profil penjual yang di simpan pada tabel penjual di *database*.

   Operasi ini terdiri dari beberapa operasi, yaitu :

   i. Melakukan *insert* pada tabel penjual
   ii. Melakukan *update* pada tabel penjual

5. Melihat informasi harga terkini
   Operasi melihat informasi harga terkini ini bertujuan untuk dapat mengetahui kondisi harga produk di pasaran secara baru.

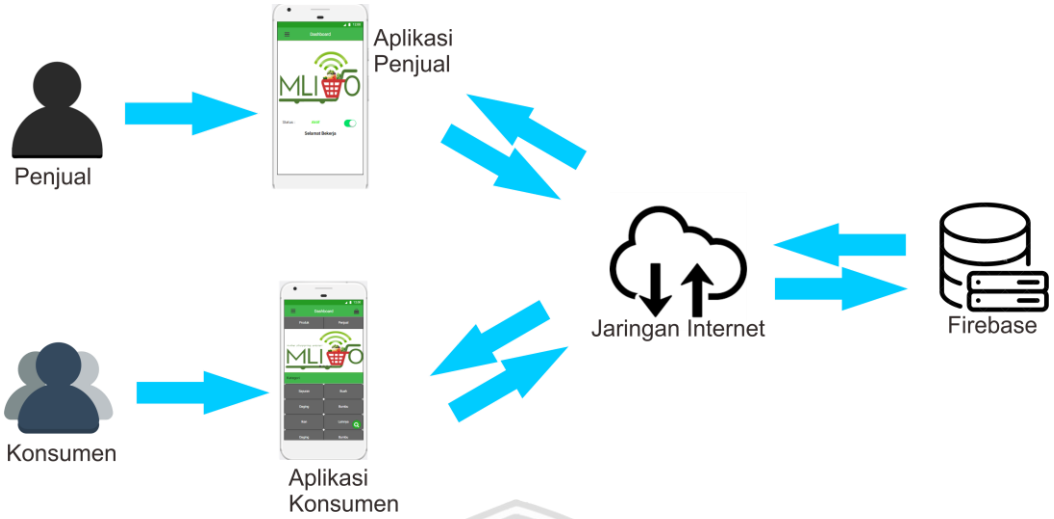
6. Mengirimkan obrolan
   Operasi ini bertujuan untuk dapat melakukan aktivitas kirim obrolan dengan konsumen.

7. Mengelola Lokasi
   Operasi ini bertujuan untuk dapat mengelola fitur lokasi pada aplikasi seperti berbagi lokasi dengan konsumen.

8. *Logout*
   Operasi ini bertujuan untuk dapat keluar dari sistem sehingga penjual tidak dapat menggunakan fungsi – fungsi sebagai penjual.

4.1.4.2 Cara Penggunaan Perangkat Lunak MLIJO

Perangkat lunak *MLIJO* memiliki langkah kerja yang terbagi menjadi 2 yaitu, pada sisi aplikasi konsumen dan aplikasi penjual. Cara kerja sistem aplikasi *MLIJO* secara umum digambarkan seperti yang ditunjukan pada Gambar 4.1 dimana untuk menjalankan aplikasi *MLIJO* ini diperlukan koneksi jaringan internet untuk dapat mengakses data yang tersimpan pada *database firebase*:



Gambar 4.1 Langkah Kerja Perangkat Lunak MLIJO

Pada gambar 4.1 diatas, digambarkan bagaimana alur sistem penggunaan aplikasi Mlijo secara umum. Dimana penjual akan melakukan operasi penggunaan aplikasi pada aplikasi Mlijo sisi penjual yang terkoneksi jaringan internet untuk melakukan transaksi data yang tersimpan pada *firebase*. Begitu pula pada sisi konsumen, dimana konsumen menjalankan aplikasi Mlijo pada sisi konsumen yang terkoneksi dengan jaringan internet untuk mengakses data yang tersimpan di dalam *firebase*. Komunikasi antara penjual dan konsumen dilakukan dengan perantara pertukaran data pada *firebase*.

4.1.5 Analisis Data

Tahap analisis data bertujuan untuk mendapatkan struktur data yang dibutuhkan pada perangkat lunak MLIJO. Struktur penyimpanan data pada perangkat lunak MLIJO disusun berdasarkan analisis data sebagai berikut :

a. Data pengguna MLIJO berupa *email* yang digunakan untuk registrasi dan *login*.

b. Data produk yang terdiri dari nama produk, kategori produk, deskripsi produk, harga produk, dan foto produk yang digunakan untuk informasi detail sebuah produk.

c. Data konsumen yang terdiri dari foto, nama, NIK, alamat, dan nomor telepon yang digunakan sebagai informasi konsumen dan informasi tujuan pengiriman produk pesanan.

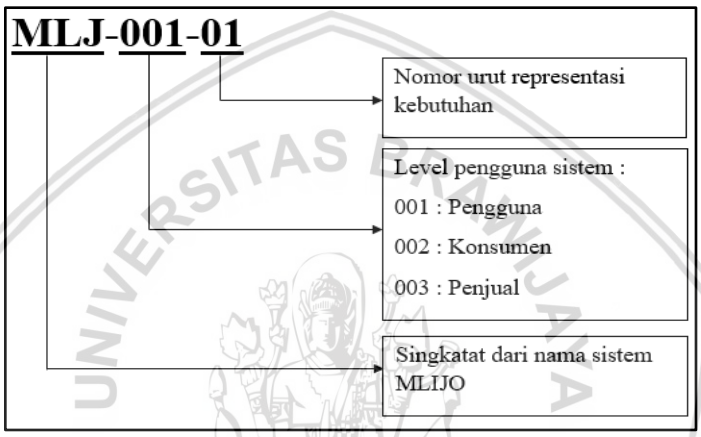
d. Data penjual yang terdiri dari foto, nama, NIK, nomor telepon, kategori produk yang dijual, area berjualan, dan waktu berjualan yang digunakan sebagai informasi detail penjual dan sebagai informasi lokasi terkini penjual.

e. Data pengguna MLIJO berupa nomor ponsel yang digunakan untuk registrasi dan *login* kedalam sistem(iterasi 0). Alasan perubahan tersebut adalah dikarenakan nomor ponsel dianggap lebih aman dalam

memverifikasi kebenaran pengguna, sebab data dari pemilik nomor ponsel saat ini telah diverifikasi oleh pemerintah sesuai dengan NIK penduduk Indonesia seperti yang tercantum dalam Peraturan Menteri Kominfo Nomor 14 Tahun 2017 (PERMENKOMINFO, 2017).

4.1.6 Spesifikasi Kebutuhan

Pada tahap ini membahas tentang daftar kebutuhan yang terdiri dari spesifikasi kebutuhan fungsional dan kebutuhan non-fungsional yang terdapat dalam sistem. Spesifikasi kebutuhan yang dituliskan merupakan spesifikasi kebutuhan hasil dari iterasi 0. Berikut ini merupakan aturan penomoran kode fungsi yang digunakan dalam spesifikasi kebutuhan fungsional pada Gambar 4.2 berikut ini:



Gambar 4.2 Aturan Penomoran Kode Fungsi

4.1.6.1 Spesifikasi Kebutuhan Fungsional

Spesifikasi kebutuhan akan dijelaskan sesuai dengan kebutuhan yang dibutuhkan oleh masing-masing aktor. Spesifikasi kebutuhan fungsional ditunjukkan pada Tabel 4.8 untuk pengguna, tabel 4.10 untuk konsumen dan 4.13 untuk penjual :

Tabel 4.8 Spesifikasi Kebutuhan Fungsional Pengguna

Kode Fungsi	Nama Fungsi	Deskripsi	*Use Case*
MLJ_001_01	Register	Sistem harus menyediakan fasilitas untuk registrasi sehingga pengguna dapat menjadi konsumen maupun penjual dan bisa menggunakan fungsionalitas sistem.	Register
MLJ_001_02	Login	Sistem harus menyediakan fasilitas login sehingga pengguna dapat masuk dan menggunakan fitur sistem.	Login

Tabel 4.9 Spesifikasi Kebutuhan Fungsional Pengguna Iterasi 0

Kode Fungsi	**Nama Fungsi**	**Deskripsi**	Keterangan
MLJ_001_03	Autentifikasi	Sistem harus menyediakan fasilitas untuk melakukan autentifikasi dimana pengguna memasukan nomor telepon untuk dapat login atau *register* ke dalam sistem.	Perubahan dari fungsi *register* dan login
MLJ_001_04	Memasukan Data Pengguna Baru	Sistem harus menyediakan fasilitas untuk melakukan input data untuk pengguna baru yang datanya belum tersimpan dalam sistem.	Fungsi baru.

Spesifikasi kebutuhan fungsional pengguna pada iterasi 0 ini terdapat perubahan fungsional. Pertama, pada fungsi MLJ_001_01 atau Register dan MLJ_001_02 atau login dihapus dan digantikan oleh fungsi MLJ_001_03 atau fungsi Autentifikasi. Hal ini dilakukan karena untuk dapat masuk ke dalam sistem tidak memerlukan lagi *register* dan login dengan *email* dan *password*, namun diubah menjadi nomor ponsel. Sehingga kedua fungsi tersebut dihapus dan digantikan oleh fungsi baru yaitu autentifikasi.

Kedua, pada fungsi MLJ_001_04 atau fungsi Memasukan Data Pengguna Baru merupakan fungsi baru yang ditambahkan untuk melengkapi fungsi yang telah ada sebelumnya.

Tabel 4.10 Spesifikasi Kebutuhan Fungsional Konsumen

Kode Fungsi	**Nama Fungsi**	**Deskripsi**	***Use Case***
MLJ_002_01	Melakukan pencarian produk	Sistem harus menyediakan fasilitas pencarian produk berdasarkan kategori atau dengan kolom pencarian	Melakukan pencarian produk
MLJ_002_02	Melakukan *filter* pencarian	Sistem harus menyediakan fasilitas untuk memfilter pencarian berdasarkan harga,kategori dan lokasi.	Melakukan *filter* pencarian
MLJ_002_03	Melihat daftar produk	Sistem harus menyediakan fasilitas untuk melihat daftar	Melihat daftar produk

		seluruh produk yang tersedia.	
MLJ_002_04	Menampilkan detail produk	Sistem harus menyediakan fasilitas untuk melihat informasi detail produk yang telah dipilih oleh konsumen.	Menampilkan detail Produk
MLJ_002_05	Melakukan pemesanan	Sistem harus menyediakan fasilitas untuk melakukan pemesanan produk kepada penjual produk yang tersedia.	Melakukan pemesanan
MLJ_002_06	Melakukan pemesanan khusus	Sistem harus menyediakan fasilitas untuk melakukan pemesanan produk yang belum tersedia dalam daftar produk kepada penjual tertentu.	Melakukan pemesanan khusus
MLJ_002_07	Melihat detail transaksi	Sistem harus menyediakan fasilitas untuk melihat informasi detail transaksi produk.	Melihat detail transaksi
MLJ_002_08	Menampilkan daftar transaksi	Sistem harus menyediakan fasilitas untuk melihat daftar transaksi.	Mengelola pembelian
MLJ_002_09	Melihat status pesanan	Sistem harus menyediakan fasilitas untuk menampilkan informasi terkait status pesanan.	Melihat status pesanan
MLJ_002_10	Melakukan konfirmasi penerimaan	Sistem harus menyediakan fasilitas untuk melakukan konfirmasi bahwa produk sudah diterima oleh konsumen.	Melakukan konfirmasi penerimaan produk
MLJ_002_11	Memberikan Ulasan	Sistem harus menyediakan fasilitas untuk membuat ulasan terhadap penjual.	Memberikan Ulasan
MLJ_002_12	Melihat riwayat transaksi	Sistem harus menyediakan fasilitas untuk menampilkan riwayat transaksi konsumen.	Melihat riwayat transaksi

MLJ_002_13	Melihat info harga terkini	Sistem harus menyediakan fasilitas untuk menampilkan informasi harga terkini.	Melihat info harga terkini
MLJ_002_14	Melihat lokasi penjual aktif	Sistem harus menyediakan fasilitas untuk menampilkan lokasi penjual aktif disekitar konsumen.	Melihat lokasi penjual aktif
MLJ_002_15	Melihat Daftar penjual	Sistem harus menyediakan fasilitas untuk melihat daftar penjual yang terdaftar dalam sistem.	Melihat Daftar penjual
MLJ_002_16	Melihat profil penjual	Sistem harus menyediakan fasilitas untuk melihat profil penjual.	Melihat profil penjual
MLJ_002_17	Melihat info gizi	Sistem harus menyediakan fasilitas untuk melihat informasi nilai gizi.	Melihat info gizi
MLJ_002_18	Mengirimkan obrolan	Sistem harus menyediakan fasilitas untuk mengirimkan obrolan kepada penjual.	Mengirimkan obrolan
MLJ_002_19	Mengelola data profil	Sistem harus menyediakan fasilitas untuk mengelola profil konsumen.	Mengelola data profil
MLJ_002_20	Logout	Sistem harus menyediakan fasilitas untuk logout yang berfungsi untuk keluar dari sistem.	Logout

Tabel 4.11 Spesifikasi Kebutuhan Fungsional Konsumen Iterasi 0

Kode Fungsi	**Nama Fungsi**	**Deskripsi**	Keterangan
MLJ_002_17	Melihat info gizi	Sistem harus menyediakan fasilitas untuk melihat informasi nilai gizi.	Fungsi dihapus
MLJ_002_21	Melakukan pembatalan pemesanan	Sistem harus menyediakan fasilitas untuk melakukan pembatalan pemesanan sebelum penjual melakukan konfirmasi terima order.	Penambahan fungsi

Spesifikasi kebutuhan fungsional konsumen pada iterasi 0 ini terdapat beberapa perubahan fungsional. Pertama, pada fungsi MLJ_002_17 atau fungsi melihat info gizi terdapat perubahan, fungsi tersebut dihapus dari sistem. Fungsi tersebut di hapus karena menurut sebagian besar konsumen (80%) fungsi tersebut kurang membantu, terlalu rumit dan secara tampilan kurang menarik. Sehingga pengembang memutuskan untuk menghapus fungsi tersebut.

Kedua, terdapat penambahan fungsi yaitu MLJ_002_21 atau fungsi melakukan pembatalan pemesanan. Fungsi tersebut di tambahkan guna melangkapi fungsi kelola pembelian yang sudah ada sebelumnya. Dimana konsumen di mungkinkan untuk melakukan pembatalan pemesanan selama transaksi pemesanan tersebut belum di konfirmasi oleh penjual.

Tabel 4.12 Spesifikasi Kebutuhan Fungsional Konsumen Iterasi 1

Kode Fungsi	**Nama Fungsi**	**Deskripsi**	Keterangan
MLJ_002_22	Menampilkan notifikasi	Sistem harus menyediakan fasilitas untuk dapat menampilkan pesan notifikasi pada kondisi tertentu.	Penambahan fungsi

Spesifikasi kebutuhan fungsional konsumen pada iterasi 1 ini terdapat satu perubahan fungsional berupa penambahan fungsi MLJ_002_22 atau fungsi menampilkan notifikasi. Fungsi tersebut di tambahkan guna melengkapi fungsi dasar dari sebuah aplikasi untuk dapat menampilkan pesan notifikasi atau pemberitahuan jika terjadi kondisi – kondisi tertentu. Fungsi tersebut merupakan hasil iterasi dari sisi konsumen, namun juga terdapat pada sisi penjual karena melibatkan dua sisi pengguna.

Tabel 4.13 Spesifikasi Kebutuhan Fungsional Penjual

Kode Fungsi	**Nama Fungsi**	**Deskripsi**	***Use Case***
MLJ_003_01	Mengelola data produk	Sistem harus menyediakan fasilitas untuk mengelola data produk yang dijual.	Mengelola data produk
MLJ_003_02	Membuat data produk	Sistem harus menyediakan fasilitas untuk membuat data produk baru.	Membuat data produk
MLJ_003_03	Merubah data produk	Sistem harus menyediakan fasilitas untuk merubah data produk yang telah ada.	Merubah data produk
MLJ_003_04	Menghapus data produk	Sistem harus menyediakan fasilitas untuk menghapus data produk yang telah ada.	Menghapus data produk

MLJ_003_05	Melihat daftar produk	Sistem harus menyediakan fasilitas untuk melihat data produk yang dijual.	Melihat daftar produk
MLJ_003_06	Mengelola penjualan	Sistem harus menyediakan fasilitas untuk menglola penjualan.	Mengelola penjualan
MLJ_003_07	Melihat Daftar pesanan baru	Sistem harus menyediakan fasilitas untuk melihat daftar pesanan produk terbaru dari konsumen.	Melihat Daftar pesanan
MLJ_003_08	Melihat detail pesanan	Sistem harus menyediakan fasilitas untuk melihat informasi detail pesanan dari setiap konsumen.	Melihat detail pesanan
MLJ_003_09	Melakukan konfirmasi pesanan	Sistem harus menyediakan fasilitas untuk melakukan konfirmasi pesanan pada sisi penjual terhadap pesanan produk konsumen.	Melakukan konfirmasi pesanan
MLJ_003_10	Melihat peta menuju lokasi konsumen	Sistem harus menyediakan fasilitas untuk melihat peta/rute menuju lokasi konsumen dari lokasi penjual berada.	Melihat peta menuju lokasi konsumen
MLJ_003_11	Melihat status penjualan	Sistem harus menyediakan fasilitas untuk melihat daftar status penjualan.	Melihat status penjualan
MLJ_003_12	Memperbarui status penjualan	Sistem harus menyediakan fasilitas untuk melakukan perubahan status penjualan.	Memperbarui status penjualan
MLJ_003_13	Melihat riwayat transaksi	Sistem harus menyediakan fasilitas untuk melihat riwayat transaksi yang telah dilakukan.	Melihat riwayat transaksi
MLJ_003_14	Mengelola aktivasi akun	Sistem harus menyediakan fasilitas untuk mengelola status aktivasi akun.	Mengelola aktivasi akun
MLJ_003_15	Mengelola data profil	Sistem harus menyediakan fasilitas untuk mengelola data profil penjual.	Mengelola data profil

MLJ_003_16	Melihat informasi harga terkini	Sistem harus menyediakan fasilitas untuk menampilkan informasi harga terkini.	Melihat informasi harga terkini
MLJ_003_17	Mengirimkan obrolan	Sistem harus menyediakan fasilitas untuk mengirim obrolan kepada konsumen.	Mengirimkan obrolan
MLJ_003_18	Mengelola lokasi	Sistem harus menyediakan fasilitas untuk mengelola pengaturan lokasi penjual.	Mengelola lokasi
MLJ_003_19	Logout	Sistem harus menyediakan fasilitas untuk logout yang berfungsi untuk keluar dari sistem.	Logout

4.1.6.2 Spesifikasi Kebutuhan Non-Fungsional

Daftar kebutuhan non-fungsional perangkat lunak MLIJO ditunjukkan pada tabel 4.8 seperti berikut :

Tabel 4.14 Spesifikasi kebutuhan Non-Fungsional

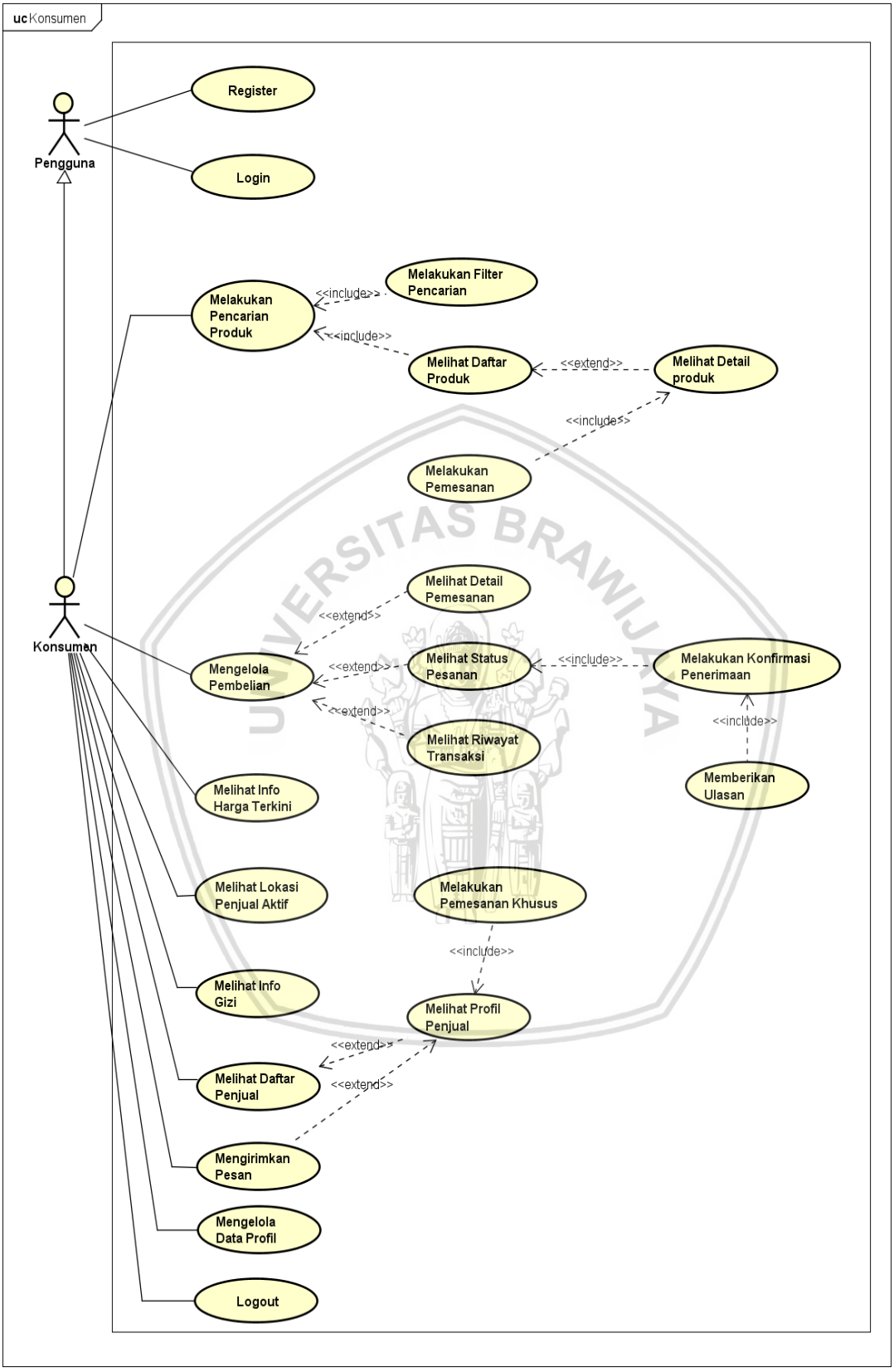
Parameter	Deskripsi Kebutuhan
Usability	Perangkat lunak harus dibuat dengan semudah mungkin sesuai dengan kebutuhan pengguna agar dapat dengan mudah digunakan oleh pengguna.
Compability	Perangkat lunak harus dapat dijalankan pada beberapa sistem operasi yang berbeda.

4.1.7 Diagram *Use Case*

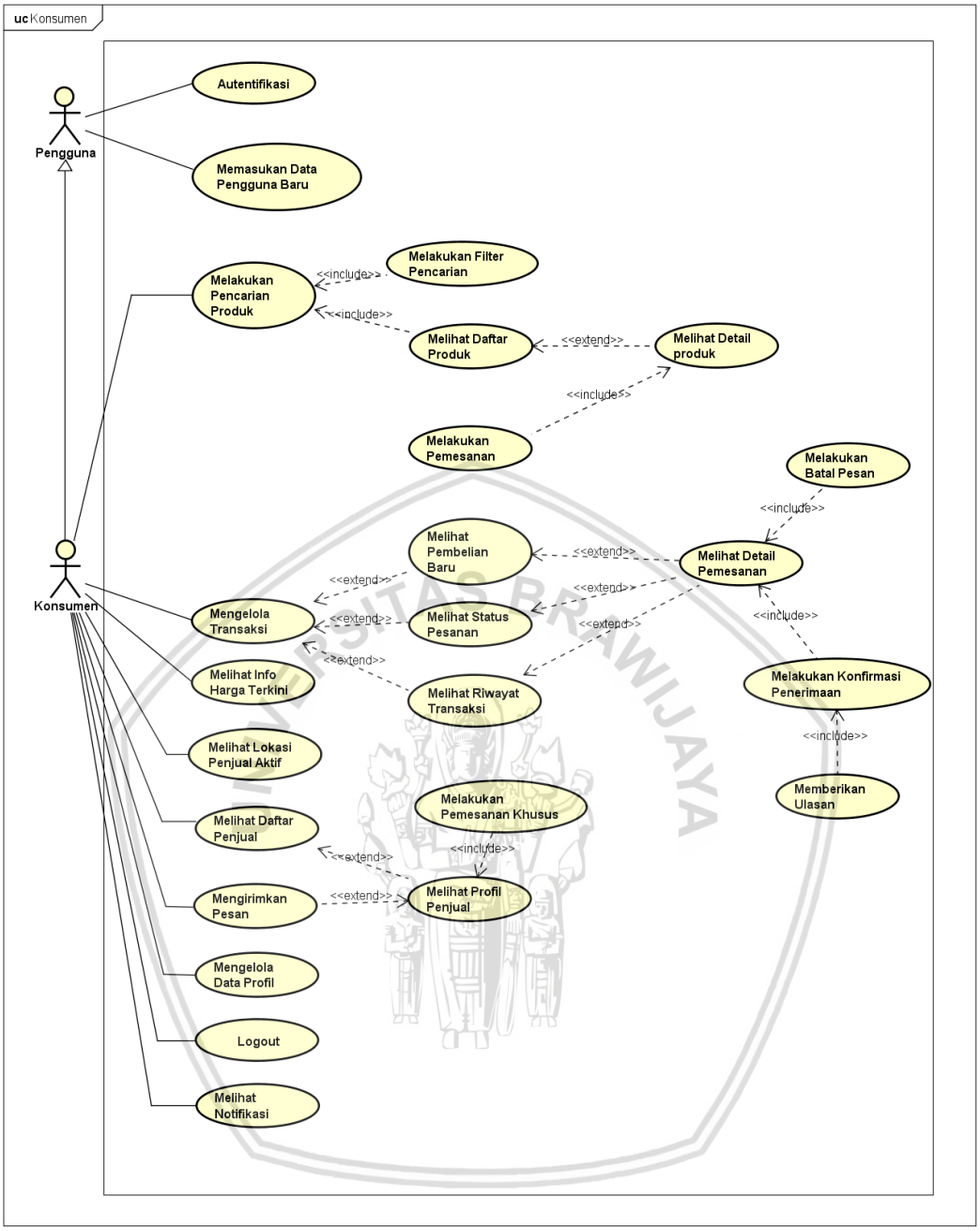
Diagram *Use Case* merupakan sebuah diagram yang digunakan sebagai penggambaran perilaku aktor dengan sistem. Tujuan dari digunakannya diagram *Use Case* adalah untuk menggambarkan seluruh kebutuhan sistem yang akan dikembangkan sehingga siapa pun yang melihat dapat memahami alur sistem tersebut berdasarkan fungsinya. Diagram *Use Case* ini dibuat berdasarkan kebutuhan fungsional sistem yang telah didapatkan analisis kebutuhan hasil dari iterasi 0 yang dibagi menjadi 2 diagram seperti berikut :

4.1.7.1 Diagram *Use Case* Aplikasi Konsumen

Diagram *Use Case* ini melibatkan konsumen sebagai aktor. Dalam *Use Case* ini konsumen didefinisikan sebagai pengguna aplikasi secara khusus yang memiliki hak akses sebagai konsumen pada aplikasi MLIJO. Diagram *Use Case* untuk konsumen ditunjukkan pada Gambar 4.3 berikut ini :



Gambar 4.3 Diagram *Use Case* Konsumen



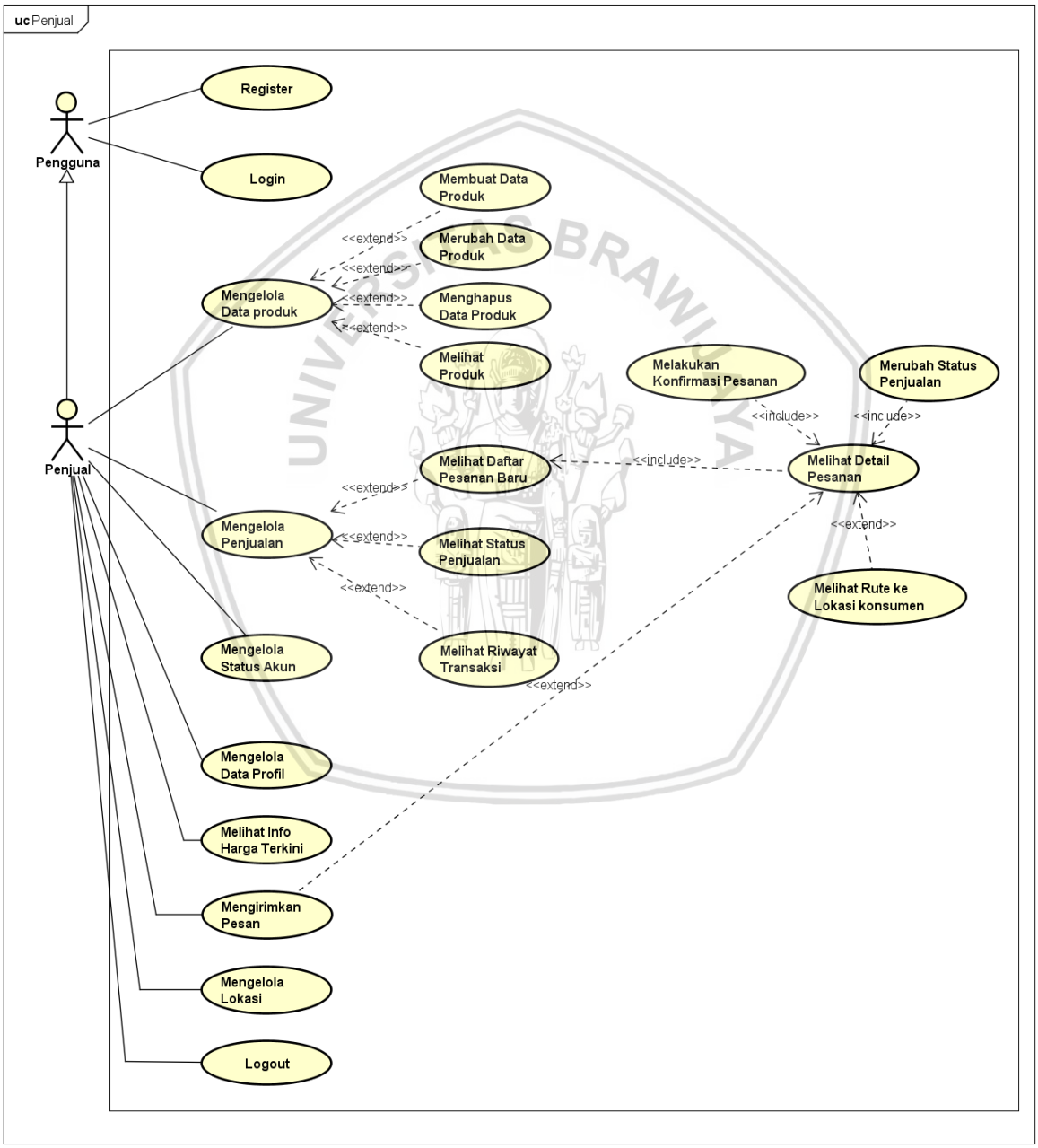
Gambar 4.4 Diagram *Use Case* Konsumen Iterasi 1

Pada diagram *Use Case* 4.4 diatas menunjukan hasil iterasi 1, dimana terjadi beberapa perubahan yaitu penambahan *Use Case* autentifikasi, input data user baru dan melakukan batal pesan. Selain itu terjadi penghapusan *Use Case register*, login dan melihat info gizi. Penambahan *Use Case* autentifikasi dan input data user baru dilakukan karena terjadi perubahan fungsi autentifikasi sistem sehingga *register* dan login tidak diperlukan lagi, sedangkan melakukan batal pesan untuk menyempurnakan fungsi kelola pembelian dimana konsumen dimungkinkan untuk melakukan pembatalan pembelian produk selama penjual belum melakukan konfirmasi penerimaan pesanan sesuai dengan hasil iterasi 1 dengan konsumen.

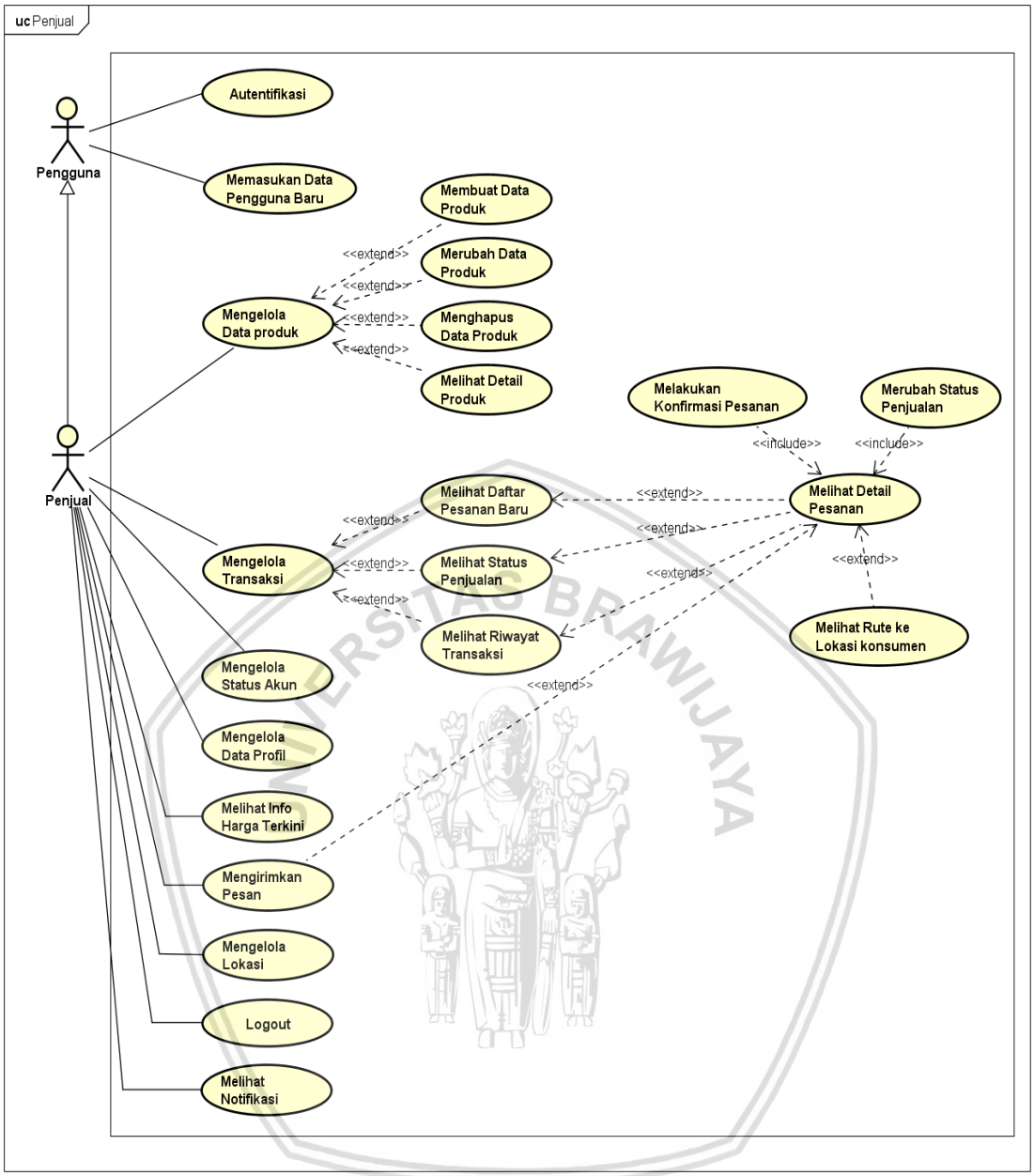
Sedangkan *Use Case* melihat info gizi di hapus karena berdasarkan hasil iterasi 1 dimana 80% konsumen menyatakan fungsi tersebut kurang membantu proses bisnis konsumen dalam menggunakan aplikasi *MLIJO* tersebut.

4.1.7.2 Diagram *Use Case* Aplikasi Penjual

Diagram *Use Case* ini melibatkan penjual sebagai aktor. Dalam *Use Case* ini penjual didefinisikan sebagai pengguna aplikasi secara khusus yang memiliki hak akses sebagai penjual produk pada aplikasi *MLIJO*. Diagram *Use Case* untuk penjual ditunjukkan pada Gambar 4.5 berikut ini :



Gambar 4.5 Diagram *Use Case* Penjual



Gambar 4.6 Diagram *Use Case* Penjual Iterasi 1

Pada diagram *Use Case* 4.6 diatas menunjukan hasil iterasi 1, dimana terjadi beberapa perubahan yaitu penambahan *Use Case* autentifikasi dan input data user baru. Selain itu terjadi penghapusan *Use Case register* dan login. Penambahan *Use Case* autentifikasi dan input data user baru dilakukan karena terjadi perubahan fungsi autentifikasi sistem sehingga *register* dan login tidak diperlukan lagi.

4.1.8 Skenario *Use Case*

Skenario *Use Case* merupakan urutan alur dan penjelasan lebih lengkap dari diagram *Use Case* yang telah didefinisikan. Sama seperti pada diagram *Use Case*, skenario *Use Case* yang di tuliskan adalah hasil dari iterasi 0.

4.1.8.1 Skenario *Use Case* Pengguna

1 Skenario *Use Case Register*

Tabel 4.15 Skenario *Use Case* Register

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_001_01
Nama	Registrasi
Tujuan	Untuk melakukan pendaftaran menjadi pelanggan pada sistem.
Deskripsi	*Use Case* ini menjelaskan bagaimana pengguna melakukan registrasi untuk melakukan pendaftaran menjadi pelanggan pada sistem.
Aktor	Pengguna
Skenario Utama	
Kondisi Awal	Sistem menampilkan form login.
Aksi Aktor	Reaksi Sistem
1. Pengguna memilih tombol registrasi pada halaman login.	2. Sistem menampilkan halaman registrasi yang terdiri dari *form input* mengenai biodata pengguna yang akan diisikan.
3. Pengguna memasukkan data registrasi lalu menekan tombol daftar.	4. Sistem akan melakukan pengecekkan terhadap data yang telah diisikan oleh pengguna. Apabila data belum ada di dalam *database* sistem, maka data akan disimpan dalam *database* sistem.
Skenario Alternatif 1 : Jika data sudah terdaftar	
	5. Sistem akan menampilkan pesan peringatan bahwa data yang dimasukkan sudah terdapat dalam *database* sistem.
Skenario Alternatif 2 : Jika data yang dimasukkan belum lengkap	
	6. Sistem akan menampilkan pesan peringatan bahwa data yang dimasukkan dalam for isian registrasi belum lengkap.

Kondisi Akhir	Sistem kan menampilkan pesan bahwa registrasi berhasil dilakukan dan menampilkan halaman login.

2 Skenario *Use Case Login*

Tabel 4.16 Skenario *Use Case* Login

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_001_02
Nama	Login
Tujuan	Agar pengguna bisa masuk kedalam sistem.
Deskripsi	*Use Case* ini menjelaskan tentang bagaimana pengguna melakukan login untuk masuk kedalam sistem sebagai pelanggan.
Aktor	Pengguna
Skenario Utama	
Kondisi Awal	Sistem menampilkan *form* login.
Aksi Aktor	Reaksi Sistem
1. Pengguna memasukkan data login seperti (email dan *password*) lalu menekan tombol Login.	2. Sistem akan melakukan pengecekkan terhadap data yang telah dimasukan oleh pengguna.
	3. Sistem mengarahkan ke halaman awal sistem.
Skenario Alternatif 1 : jika email dan *password* kosong	
	4. Sistem akan menampilkan pesan peringatan bahwa email atau *password* belum dimasukkan.
Skenario Alternatif 2 : jika email dan *password* tidak terdaftar	
	5. Sistem akan menampilkan pesan peringatan bahwa emal tidak terdaftar dalam sistem.

Kondisi Akhir	Sistem menampilkan halaman utama yaitu halaman *dashboard*.

3 Skenario *Use Case* Autentifikasi

Tabel 4.17 Skenario *Use Case* Autentifikasi

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_001_03
Nama	Autentifikasi
Tujuan	Untuk melakukan pendaftaran atau login kedalam pada sistem.
Deskripsi	*Use Case* ini menjelaskan bagaimana pengguna melakukan registrasi atau login untuk dapat menggunakan fungsionalita sistem.
Aktor	Pengguna
Skenario Utama	
Kondisi Awal	Pengguna dalam kondisi belum terdaftar. Sistem menampilkan form input nomor ponsel.
Aksi Aktor	Reaksi Sistem
1. Pengguna memasukan data berupa nomor ponsel dan tekan tombol selanjutnya.	2. Sistem menerima input pengguna dan mengeksekusi perintah minta kode verifikasi. Halaman berubah ke input kode verifikasi.
	3. Sistem mengirim kode verifikasi dalam bentuk sms.
4. Pengguna memasukkan kode verifikasi kedalam kolom yang tersedia.	5. Sistem akan melakukan pengecekkan terhadap data yang telah diisikan oleh pengguna. Apabila kode verifikasi benar maka akan masuk ke halaman input data user baru.
Skenario Alternatif 1 : Jika data sudah terdaftar	
	6. Sistem akan melakukan pengecekkan terhadap data yang telah diisikan oleh

	pengguna. Apabila kode verifikasi benar maka akan masuk ke halaman *dashboard*.
Kondisi Akhir	Sistem menampilkan halaman input data user baru atau halaman *dashboard*.

4 Skenarion *Use Case* Memasukan Data Pengguna Baru

Tabel 4.18 Skenario *Use Case* Memasukan Data Pengguna Baru

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_001_04
Nama	Memasukan Data Pengguna baru
Tujuan	Untuk melakukan input detail data pengguna baru pada sistem.
Deskripsi	*Use Case* ini menjelaskan bagaimana pengguna melakukan input data untuk melengkapi data pengguna pada sistem.
Aktor	Pengguna
Skenario Utama	
Kondisi Awal	Sistem menampilkan form input data pengguna baru.
Aksi Aktor	Reaksi Sistem
1. Pengguna memasukan data.	
2. Pengguna menekan tombol simpan.	3. Sistem menerima input pengguna dan data yang telah diisikan oleh pengguna akan disimpan dalam *database* sistem.
Kondisi Akhir	Sistem akan menampilkan halaman *dashboard*.

4.1.8.2 Skenario *Use Case* Aplikasi Konsumen

1 **Skenario *Use Case Melakukan Pencarian Produk***

Tabel 4.19 Skenario *Use Case* Melakukan Pencarian Produk

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_01

Nama	Melakukan Pencarian Produk
Tujuan	Untuk melakukan pencarian produk
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat melakukan pencarian produk berdasarkan kategori atau dengan memasukkan kata kunci pada kolom pencarian.
Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet. Sistem menampilkan halaman awal aplikasi.
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih kategori produk yang diinginkan	2. Sistem menerima data berupa kategori produk yang dipilih dan akan menampilkan daftar produk dalam bentuk *listview*.
Skenario Alternatif 1 : Konsumen memasukkan kata kunci pada kolom pencarian	
3. Konsumen memilih icon pencarian	4. Sistem menampilkan kolom pencarian berupa text input
5. Konsumen melakukan input kata kunci pada kolom pencarian	6. Sistem menerima data berupa kata kunci yang di masukkan pada kolom pencarian dan akan menampilkan produk yang mengandung kata kunci tersebut dalam bentuk *listview*
Kondisi Akhir	Daftar produk berhasil ditampilkan dalam bentuk *listview* sesuai dengan pencarian.

2 **Skenario *Use Case Melakukan Filter Pencarian***

Tabel 4.20 Skenario *Use Case* Melakukan Filter Pencarian

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_02
Nama	Melakukan Filter Pencarian
Tujuan	Untuk melakukan filter terhadap hasil pencarian

Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat melakukan filter terhadap hasil pencarian berdasarkan kategori dan harga
Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet. Konsumen telah melakukan pencarian produk
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih menu filter	2. Sistem menampilkan menu pilihan filter berdasarkan kategori dan harga
3. Konsumen memasukan pilihan filter yang dinginkan	4. Sistem menerima data inputan dan menampilkan hasil penyaringan data berupa list view
Kondisi Akhir	Daftar produk berhasil ditampilkan dalam bentuk *listview* sesuai dengan pencarian.

3 **Skenario *Use Case* Melihat Daftar Produk**

Tabel 4.21 Skenario *Use Case* Melihat Daftar Produk

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_03
Nama	Melihat Daftar Produk
Tujuan	Untuk dapat melihat semua produk terdaftar
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat melihat daftar dari semua produk terdaftar.
Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet. Konsumen telah melakukan pencarian produk.
Aksi Aktor	Reaksi Sistem

1. Konsumen memilih menu kategori	2. Sistem menampilkan daftar produk terdaftar
Kondisi Akhir	Sistem menampilkan daftar produk dalam *listview*.

4 **Skenario *Use Case Menampilkan Detail Produk***

Tabel 4.22 Skenario *Use Case* Menampilkan Detail Produk

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_04
Nama	Menampilkan Detail Produk
Tujuan	Untuk menampilkan detail produk terpilih
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat menampilkan detail produk yang dipilih
Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet. Konsumen berada pada halaman daftar produk.
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih produk yang dinginkan	2. Sistem menampilkan detail produk yang dipilih
Kondisi Akhir	Detail produk berhasil ditampilkan dalam bentuk halaman *activity*.

5 **Skenario *Use Case Melakukan Pemesanan Produk***

Tabel 4.23 Skenario *Use Case* Memesan Produk

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_05
Nama	Melakukan pemesanan produk
Tujuan	Untuk melakukan pemesanan produk
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat melakukan pemesanan produk yang terdapat dalam daftar produk

Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet. Konsumen berada dalam halaman detail produk yang dipilih
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih menu pesan	2. Sistem menerima input konsumen dan melanjutkan ke halaman pesan produk.
3. Konsumen melakukan input data dan kemudian menekan tombol pesan	4. Sistem menerima input konsumen dan menyimpan pesanan konsumen kedalam *database*.
Kondisi Akhir	Data pesanan telah tersimpan dan konsumen kembali ke halaman *dashboard*.

6 **Skenario *Use Case Memesan Produk Khusus***

Tabel 4.24 Skenario *Use Case* Memesan Produk Khusus

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_06
Nama	Memesan Produk khusus
Tujuan	Untuk melakukan pemesan produk khusus
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat melakukan pemesanan produk khusus yang tidak terdapat dalam daftar produk.
Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet. Konsumen berada pada halaman detail penjual yang diinginkan.
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih menu pesan produk	2. Sistem menampilkan form input data produk.

khusus pada halaman profil penjual	
3. Konsumen memilih tombol pesan	4. Sistem menerima input konsumen dan menyimpan pesanan konsumen kedalam *database*.
Kondisi Akhir	Data pesanan telah tersimpan dan konsumen kembali ke halaman *dashboard*.

7 **Skenario *Use Case* Mengelola Pembelian**

Tabel 4.25 Skenario *Use Case* Mengelola Pembelian

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_07
Nama	Mengelola Pembelian
Tujuan	Untuk dapat melihat menu mengelola pembelian
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat melihat menu kelola pembelian.
Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen dalam keadaan session login.
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih menu drawer	2. Sistem menampilkan halaman drawer
3. Konsumen memilih menu kelola pembelian	4. Sistem menampilkan halaman kelola pembelian.
Kondisi Akhir	Sistem menampilkan halaman kelola pembelian.

8 **Skenario *Use Case Melihat Detail Pemesanan***

Tabel 4.26 Skenario *Use Case* Melihat Detail Pemesanan

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_08
Nama	Melihat Detail Pemesanan

Tujuan	Untuk melihat detail dari pesanan konsumen
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat melihat detail dari pesanan yang telah dilakukan
Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet. Konsumen telah melakukan pemesanan dan berada pada halaman kelola pembelian.
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih salah satu menu pada halaman kelola pembelian	2. Sistem menampilkan daftar pesanan dalam bentuk *listview*.
3. Konsumen memilih pesanan yang ingin dilihat.	4. Sistem menampilkan detail pesanan konsumen dalam halaman *activity*.
Kondisi Akhir	Detail pesanan ditampilkan dalam bentuk halaman *activity*.

9 **Skenario *Use Case Menampilkan Status Pesanan***

Tabel 4.27 Skenario *Use Case* Menampilkan Status Pesanan

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_09
Nama	Menampilkan Status pesanan
Tujuan	Untuk menampilkan status pesanan konsumen
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat menampilkan status pesanan terhadap pesanan yang telah di buat.
Aktor	Konsumen
Skenario Utama	

Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet. Konsumen berada pada halaman kelola pembelian
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih menu status pesanan	2. Sistem menampilkan daftar pesanan dalam bentuk *listview* beserta status pesanan terkini
Kondisi Akhir	Status pesanan ditampilkan dalam bentuk *listview*.

10 **Skenario *Use Case Melakukan Konfirmasi Penerimaan Pesanan***

Tabel 4.28 Skenario *Use Case* Melakukan Konfirmasi Penerimaan Pesanan

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_10
Nama	Melakukan konfirmasi penerimaan pesanan
Tujuan	Untuk melakukan konfirmasi bahwa pesanan telah di terima
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat melakukan konfirmasi penerimaan pesanan jika telah menerima produk yang di pesan.
Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet. Konsumen berada pada halaman status pesanan.
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih produk pesanan yang telah dikirim pada tabel status pesanan.	2. Sistem menerima input dari konsumen dan menampilkan detail pesanan beserta tombol terima.

3. Konsumen memilih tombol terima.	4. Sistem menerima input dari konsumen dan merubah data status pesanan pada *database* menjadi diterima.
Kondisi Akhir	Status pesanan berubah menjadi diterima.

11 **Skenario *Use Case Membuat Ulasan***

Tabel 4.29 Skenario *Use Case* Memberikan Ulasan

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_11
Nama	Memberikan ulasan
Tujuan	Untuk membuat ulasan terhadap pesanan yang telah terkirim
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat membuat ulasan untuk produk pesanan yang telah dikirim oleh penjual.
Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet. Konsumen telah melakukan konfirmasi penerimaan dan berada pada halaman detail pesanan.
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih menu buat ulasan.	2. Sistem menerima input dari konsumen kemudian menampilkan kolom ulasan berupa text input dan penilaian bintang.
3. Konsumen melakukan input pada kolom text input dan kolom penilaian bintang	
4. Konsumen memilih tombol kirim	5. Sistem menyimpan input konsumen dan kembali kehalaman daftar pesanan.
Kondisi Akhir	Sistem menyimpan data ulasan dan kembali ke halaman daftar pesanan.

12 **Skenario *Use Case* Melihat Riwayat Transaksi**

Tabel 4.30 Skenario *Use Case* Melihat Riwayat Transaksi

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_12
Nama	Melihat Riwayat Transaksi
Tujuan	Untuk melihat riwayat dari transaksi
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat melihat informasi riwayat transaksi yang telah dilakukan.
Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet. Konsumen berada pada halaman kelola pembelian
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih menu riwayat transaksi	2. Sistem menampilkan daftar transaksi yang pernah dilakukan dalam bentuk *listview*
Kondisi Akhir	Informasi riwayat transaksi di tampilkan dalam bentuk *listview* beserta status akhir.

13 **Skenario *Use Case Melihat Informasi Harga Terkini***

Tabel 4.31 Skenario *Use Case* Melihat Informasi Harga Terkini

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_13
Nama	Melihat informasi harga terkini
Tujuan	Untuk melihat informasi harga produk terbaru
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat melihat informasi harga terkini yang terhubung dengan situs siskaperbapo
Aktor	Konsumen
Skenario Utama	

Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet. Konsumen berada pada halaman drawer
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih menu info harga	2. Sistem menampilkan daftar harga produk kebutuhan pasar terbaru dalam bentuk tabel
Kondisi Akhir	Informasi harga terkini di tampilkan dalam bentuk tabel

14 **Skenario *Use Case Melihat Lokasi Penjual Aktif***

Tabel 4.32 Skenario *Use Case* Melihat Lokasi Penjual Aktif

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_14
Nama	Melihat Lokasi Penjual Aktif
Tujuan	Untuk dapat melihat lokasi penjual yang sedang aktif
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat melihat lokasi penjual yang dalam kondisi aktif atau sedang berjualan disekitar lokasi konsumen.
Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen telah dalam keadaan login, terkoneksi dengan jaringan internet dan GPS dalam kondisi aktif. Konsumen berada pada halaman drawer
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih menu penjual aktif	2. Sistem menerima hasil input konsumen dan menampilkan lokasi penjual aktif dalam bentuk peta.
Skenario alternatif 1 : Jika tidak terdapat penjual pada lokasi konsumen	

	3. Sistem menampilkan pesan tidak terdapat penjual di sekitar
Kondisi Akhir	Lokasi penjual aktif ditampilkan dalam bentuk peta dan ikon yang dapat di klik.

15 **Skenario *Use Case* Melihat Daftar Penjual**

Tabel 4.33 Skenario *Use Case* Melihat Daftar Penjual

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_15
Nama	Melihat Daftar penjual
Tujuan	Untuk melihat daftar dari penjual
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat melihat daftar dari seluruh penjual terdaftar.
Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet. Konsumen berada pada halaman *dashboard*.
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih tab penjual.	2. Sistem menampilkan halaman daftar penjual dalam bentuk *listview*
Kondisi Akhir	Daftar penjual ditampilkan dalam bentuk *listview*.

16 **Skenario *Use Case Melihat Profil Penjual***

Tabel 4.34 Skenario *Use Case* Melihat Profil Penjual

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_16
Nama	Melihat profil penjual
Tujuan	Untuk melihat profil dari penjual
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat melihat profil dari setiap penjual.

Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet. Konsumen berada pada halaman daftar penjual.
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih penjual yang ingin dilihat pada daftar penjual.	2. Sistem menampilkan detail profil penjual pada halaman detail penjual.
Skenario alternatif 1: Jika konsumen berada pada halaman detail produk	
3. Konsumen menekan foto penjual.	4. Sistem menampilkan detail profil penjual pada halaman detail penjual.
Kondisi Akhir	Detail profil penjual ditampilkan dalam bentuk activity.

17 **Skenario *Use Case* Melihat Informasi Gizi**

Tabel 4.35 Skenario *Use Case* Melihat Informasi Gizi

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_18
Nama	Melihat informasi Gizi
Tujuan	Untuk dapat melihat informasi kecukupan gizi
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat melihat informasi kebutuhan kecukupan nilai gizi.
Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet.
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih menu drawer	2. Sistem menampilkan halaman drawer

3. Konsumen memilih menu info gizi	4. Sistem menampilkan halaman informasi gizi dalam bentuk tabel AKG.
Kondisi Akhir	Sistem menampilkan halaman informasi gizi dalam bentuk tabel AKG

18 **Skenario *Use Case* Mengirimkan Obrolan**

Tabel 4.36 Skenario *Use Case* Mengirimkan Obrolan

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_18
Nama	Melakukan Chat
Tujuan	Untuk dapat mengirim obrolan kepada penjual
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat mengirim obrolan kepada penjual.
Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet. Konsumen dalam halaman detail penjual.
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih tombol hubungi penjual	2. Sistem menampilkan halaman obrolan berupa text input
3. Konsumen memilih tombol kirim	4. Sistem menerima input dan mengirim obrolan kepada penjual.
Skenario alternatif 1: Jika konsumen berada pada halaman detail produk.	
5. Konsumen memilih kirim pesan	6. Sistem menampilkan halaman obrolan berupa text input
Skenario alternatif 2: Jika konsumen sudah pernah mengirim obrolan kepada penjual.	
7. Konsumen memilih menu obrolan pada drawer	8. Sistem menampilkan halaman daftar obrolan.

9. Konsumen memilih penjual yang ingin dikirimi pesan	10. Sistem menampilkan halaman obrolan berupa text input.
Kondisi Akhir	Pesan terkirim kepada penjual dan sistem menampilkan pesan pada activity tersebut.

19 **Skenario *Use Case Mengelola Data Profil***

Tabel 4.37 Skenario *Use Case* Mengelola Data Profil

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_19
Nama	Mengelola data profil
Tujuan	Untuk dapat mengelola data profil konsumen
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat melakukan pengelolaan data profil (insert dan update)
Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet. Konsumen berada pada halaman pengaturan.
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih menu kelola profil	2. Sistem menampilkan detail data profil berupa kolom foto dan text input
3. Konsumen melakukan input pada kolom yang disediakan.	
4. Konsumen memilih tombol simpan	5. Sistem menerima data input konsumen dan menyimpan kedalam *database*
Kondisi Akhir	Data profil konsumen berhasil dirubah.

20 **Skenario *Use Case Logout***

Tabel 4.38 Skenario *Use Case* Logout

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_20

Nama	Logout
Tujuan	Untuk dapat keluar dari sistem
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat melakukan logout untuk keluar dari sistem.
Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen dalam keadaan session login dan terkoneksi jaringan internet.
Aksi Aktor	Reaksi Sistem
1. Konsumen memilih menu drawer	2. Sistem menampilkan halaman drawer
3. Konsumen memilih menu logout	4. Sistem meneriman masukkan konsumen dan menghapus session login konsumen.
Kondisi Akhir	Sistem menampilkan halaman login.

21. **Skenario *Use Case* Melakukan Batal Pesan**

Tabel 4.39 Skenario *Use Case* Melakukan Batal Pesan Iterasi 0

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_21
Nama	Menampilkan Batal Pesan
Tujuan	Untuk melakukan batal pesan
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat melakukan batal pesan terhadap pesanan yang telah dibuat, selama pesanan tersebut belum di konfirmasi oleh penjual
Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet. Konsumen berada pada halaman kelola pembelian
Aksi Aktor	Reaksi Sistem

1. Konsumen memilih menu pesanan baru	2. Sistem menampilkan daftar pesanan dalam bentuk *listview* beserta status pesanan terkini
3. Konsumen memilih pesanan yang ingin dibatalkan	4. Sistem menampilkan detail pesanan terpilih
5. Konsumen memilih menu batal pesan	6. Sistem menerima input dan merubah status pesanan.
Kondisi Akhir	Status pesanan berubah yang ditampilkan dalam bentuk *listview* dan berpindah ke daftar riwayat transaksi.

22. **Skenario *Use Case Melihat notifikasi***

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_002_22
Nama	Melihat notifikasi
Tujuan	Untuk melakukan lihat notifikasi
Deskripsi	*Use Case* ini menjelaskan bagaimana konsumen dapat melakukan fungsi lihat notifikasi apabila terdapat pemberitahuan seperti pesanan di tolak ataupun terdapat obrolan baru dari penjual.
Aktor	Konsumen
Skenario Utama	
Kondisi Awal	Konsumen telah dalam keadaan login dan terkoneksi dengan jaringan internet. Konsumen telah melakukan pembelian produk atau pernah mengirim obrolan terhadap penjual
Aksi Aktor	Reaksi Sistem
	1. Sistem menampilkan notifikasi pada layar perangkat lunak konsumen
2. Konsumen memilih notifikasi	3. Sistem menampilkan halaman sesuai dengan isi notifikasi

Kondisi Akhir	Status pesanan berubah yang ditampilkan dalam bentuk *listview* dan berpindah ke daftar riwayat transaksi.

4.1.8.3 Skenario *Use Case* Aplikasi Penjual

1 **Skenario *Use Case Mengelola Data Produk***

Tabel 4.40 Skenario *Use Case* Mengelola Data Produk

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_01
Nama	Mengelola data produk
Tujuan	Untuk masuk halaman kelola data produk yang dijual
Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat melakukan kelola produk yang dijual.
Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah dalam keadaan login dan terkoneksi dengan jaringan internet. Penjual berada pada halaman drawer
Aksi Aktor	Reaksi Sistem
1. Penjual memilih menu kelola produk	2. Sistem menerima input dari penjual dan menampilkan halaman kelola produk.
Kondisi Akhir	Penjual berhasil masuk halaman kelola produk.

2 **Skenario *Use Case* Membuat Data Produk**

Tabel 4.41 Skenario *Use Case* Membuat Data Produk

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_02
Nama	Membuat Data Produk
Tujuan	Untuk dapat membuat data produk baru
Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat melakukan buat data produk baru.

Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah dalam keadaan login dan terkoneksi dengan jaringan internet. Penjual dalam halaman kelola produk
Aksi Aktor	Reaksi Sistem
1. Penjual memilih icon tambah	2. Sistem menerima input dari penjual dan menampilkan form untuk input data produk
3. Penjual melakukan input pada kolom yang tersedia	
4. Penjual memlih tombol simpan	5. Sistem meneriman input dan menyimpan data input kedalam *database* produk
Kondisi Akhir	Sistem menyimpan input penjual dan menampilkan pada daftar kelola produk.

3 **Skenario *Use Case* Merubah Data Produk**

Tabel 4.42 Skenario *Use Case* Merubah Data Produk

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_03
Nama	Merubah Data Produk
Tujuan	Untuk dapat merubah data produk
Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat merubah data produk pada daftar produk.
Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah dalam keadaan login dan terkoneksi dengan jaringan internet. Penjual dalam halaman kelola produk
Aksi Aktor	Reaksi Sistem
1. Penjual memilih icon config pada	2. Sistem menampilkan menu dalam bentuk pop-up

produk yang ingin dirubah	
3. Penjual memilih menu ubah	4. Sistem menampilkan menampilkan form untuk input data produk.
5. Penjual melakukan input data dan pilih tombol simpan	6. Sistem meneriman input dan menyimpan perubahan data input kedalam *database* produk.
Kondisi Akhir	Sistem menyimpan input penjual dan menampilkan produk pada daftar produk.

4 **Skenario *Use Case* Menghapus Data Produk**

Tabel 4.43 Skenario *Use Case* Menghapus Data Produk

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_04
Nama	Menghapus
Tujuan	Untuk dapat menghapus data produk dari daftar produk
Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat melakukan hapus data produk.
Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah dalam keadaan login dan terkoneksi dengan jaringan internet. Penjual dalam halaman kelola produk
Aksi Aktor	Reaksi Sistem
1. Penjual memilih icon config pada produk yang ingin dihapus	2. Sistem menampilkan menu dalam bentuk pop-up
3. Penjual memilih menu hapus	4. Sistem menerima input penjual dan menghapus data produk dari *database*.
Kondisi Akhir	Sistem menampilkan daftar produk pada halaman kelola produk.

5 **Skenario *Use Case Melihat Detail Produk***

Tabel 4.44 Skenario *Use Case* Melihat Detail Produk

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_05
Nama	Melihat detail produk
Tujuan	Untuk melihat detail produk terpilih
Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat melihat detail dari produk yang dijual.
Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah dalam keadaan login dan terkoneksi dengan jaringan internet. Penjual berada pada halaman kelola produk
Aksi Aktor	Reaksi Sistem
1. Penjual memilih produk	2. Sistem menampilkan detail produk pada halaman activity
Kondisi Akhir	Sistem menampilkan detail produk pada halaman activity

6 **Skenario *Use Case* Mengelola Penjualan**

Tabel 4.45 Skenario *Use Case* Mengelola Penjualan

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_06
Nama	Mengelola data penjualan
Tujuan	Untuk mengelola data penjualan produk
Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat mengelola data penjualan produk
Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah dalam keadaan login dan terkoneksi dengan jaringan internet. Penjual berada pada halaman drawer

Aksi Aktor	Reaksi Sistem
1. Penjual memilih menu kelola penjualan.	2. Sistem akan menerima input penjual dan akan menampilkan ringkasan notifikasi penjualan.
Kondisi Akhir	Sistem menampilkan ringkasan status notifikasi pada halaman kelola penjualan.

7 **Skenario *Use Case Melihat Daftar Pesanan Baru***

Tabel 4.46 Skenario *Use Case* Melihat Daftar Pesanan Baru

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_07
Nama	Melihat daftar pesanan baru
Tujuan	Untuk melihat daftar pesanan produk baru
Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat melihat semua daftar pesanan produk baru yang belum diproses.
Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah dalam keadaan login dan terkoneksi dengan jaringan internet. Penjual berada pada halaman kelola penjualan
Aksi Aktor	Reaksi Sistem
1. Penjual memilih menu pesanan baru	2. Sistem menampilkan daftar pesanan dalam bentuk *listview*
Kondisi Akhir	Sistem menampilkan daftar pesanan dalam bentuk *listview*

8 **Skenario *Use Case Melihat Informasi Detail Pesanan***

Tabel 4.47 Skenario *Use Case* Melihat Informasi Detail Pesanan

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_08
Nama	Melihat informasi detail pesanan
Tujuan	Untuk menampilkan detail pesanan terpilih

Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat menampilkan detail pesanan yang dipilih
Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah dalam keadaan login dan terkoneksi dengan jaringan internet. Penjual berada pada halaman daftar pesanan baru
Aksi Aktor	Reaksi Sistem
1. Penjual memilih pesanan	2. Sistem menampilkan detail pesanan yang dipilih
Kondisi Akhir	Detail pesanan berhasil ditampilkan.

9 **Skenario *Use Case Melakukan Konfirmasi Pesanan***

Tabel 4.48 Skenario *Use Case* Melakukan Konfirmasi Pesanan

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_09
Nama	Melakukan konfirmasi pesanan
Tujuan	Untuk melakukan konfirmasi pesanan
Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat melakukan konfirmasi terhadap pesanan konsumen
Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah dalam keadaan login dan terkoneksi dengan jaringan internet. Penjual berada pada halaman detail pesanan
Aksi Aktor	Reaksi Sistem
1. Penjual memilih menu terima order	2. Sistem menerima input penjual dan merubah status pesanan menjadi sedang di proses pada *database*
Skenario Alternatif 1 : Jika penjual menolak order	
3. Penjual memilih menu tolak order	4. Sistem menerima input penjual dan merubah status pesanan menjadi pesanan

	di tolak pada *database* juga mengirim pesan kepada konsumen
Kondisi Akhir	Pesanan terkonfirmasi dan status pesanan berubah.

10 **Skenario *Use Case Melihat Peta Menuju Lokasi Konsumen***

Tabel 4.49 Skenario *Use Case* Melihat Peta Menuju Lokasi Konsumen

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_10
Nama	Melihat peta menuju lokasi konsumen
Tujuan	Untuk melihat rute menuju konsumen
Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat melihat rute jalan menuju lokasi pemesan produk
Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah dalam keadaan login, terkoneksi dengan jaringan internet dan GPS aktif. Penjual berada pada halaman detail pesanan.
Aksi Aktor	Reaksi Sistem
1. Penjual memilih menu lihat peta	2. Sistem menampilkan rute berupa peta jalan dari lokasi penjual saat ini menuju lokasi konsumen
Kondisi Akhir	Ditampilkan rute berupa peta jalan dari lokasi penjual saat ini menuju lokasi konsumen

11 **Skenario *Use Case* Melihat Status Penjualan**

Tabel 4.50 Skenario *Use Case* Melihat Status Penjualan

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_11
Nama	Melihat Status Penjualan
Tujuan	Untuk dapat melihat status dari pesanan yang telah diproses

Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat dapat melihat status dari pesanan yang telah diproses.
Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah dalam keadaan login dan terkoneksi dengan jaringan internet. Penjual dalam halaman kelola penjualan
Aksi Aktor	Reaksi Sistem
1. Penjual memilih menu status pesanan	2. Sistem menampilkan daftar status pesanan dalam bentuk *listview*
Kondisi Akhir	Sistem menampilkan daftar status pesanan dalam bentuk *listview*.

12 **Skenario *Use Case* Memperbarui Status Penjualan**

Tabel 4.51 Skenario *Use Case* Memperbarui Status Pejualan

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_12
Nama	Memperbarui Status Pesanan
Tujuan	Untuk dapat memperbarui status dari pesanan
Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat memperbarui status dari pesanan.
Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah dalam keadaan login dan terkoneksi dengan jaringan internet. Penjual dalam halaman status pesanan
Aksi Aktor	Reaksi Sistem
1. Penjual memilih pesanan yang dinginkan	2. Sistem menampilkan detail pesanan dan tombol perbarui status.

3. Penjual memilih tombol perbarui status transaksi	4. Sistem menerima input penjual dan merubah status dari pesanan terpilih pada *database*.
Kondisi Akhir	Sistem menampilkan status terbaru dari pesanan pada halaman status pesanan.

13 **Skenario *Use Case* Melihat Riwayat Transaksi**

Tabel 4.52 Skenario *Use Case* Melihat Riwayat Transaksi

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_13
Nama	Melihat Riwayat Transaksi
Tujuan	Untuk dapat melihat daftar riwayat transaksi yang pernah dilakukan
Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat melihat daftar riwayat transaksi yang pernah dilakukan.
Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah dalam keadaan login dan terkoneksi dengan jaringan internet. Penjual dalam halaman kelola penjualan
Aksi Aktor	Reaksi Sistem
1. Penjual memilih menu riwayat transaksi	2. Sistem menampilkan daftar riwayat transaksi dalam bentuk *listview*
Kondisi Akhir	Sistem menampilkan daftar riwayat transaksi dalam bentuk *listview*

14 **Skenario *Use Case Mengelola Status Akun***

Tabel 4.53 Skenario *Use Case* Mengelola Status Akun

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_14
Nama	Mengelola status akun

Tujuan	Untuk dapat merubah status aktivasi akun penjual
Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat melakukan perubahan status penjual terikini, dimana penjual tersebut sedang aktif atau menerima order maupun non-aktif atau tidak menerima order.
Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah dalam keadaan login dan terkoneksi dengan jaringan internet. Penjual berada pada halaman *Dashboard*
Aksi Aktor	Reaksi Sistem
1. Penjual menekan tombol switch aktif/non-aktif	2. Sistem menerima input penjual dan akan merubah status akun penjual
Kondisi Akhir	Sistem menerima input penjual dan akan merubah status akun penjual

15 **Skenario *Use Case Mengelola Data Profil***

Tabel 4.54 Skenario *Use Case* Mengelola Data Profil

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_15
Nama	Mengelola data profil penjual
Tujuan	Untuk dapat mengelola data profil penjual
Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat melakukan pengelolaan data profil (insert, update atau delete)
Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah dalam keadaan login dan terkoneksi dengan jaringan internet. Penjual berada pada halaman pengaturan
Aksi Aktor	Reaksi Sistem

1. Penjual memilih menu kelola profil	2. Sistem menampilkan detail data profil berupa tabel dan kolom serta checkbox kategori produk yang dijual
3. Penjual merubah data yang dinginkan	
4. Penjual memilih menu simpan	5. Sistem menerima data input penjual dan menyimpan kedalam *database*.
Kondisi Akhir	Data profil penjual berhasil diperbarui.

16 **Skenario *Use Case Melihat Informasi Harga Terkini***

Tabel 4.55 Skenario *Use Case* Melihat Informasi Harga Terkini

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_16
Nama	Melihat informasi harga terkini
Tujuan	Untuk melihat informasi harga produk terbaru
Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat melihat informasi harga terkini yang terhubung dengan situs siskaperbapo
Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah dalam keadaan login dan terkoneksi dengan jaringan internet. Penjual berada pada halaman drawer.
Aksi Aktor	Reaksi Sistem
1. penjual memilih menu info harga	2. Sistem menampilkan daftar harga produk kebutuhan pasar terbaru dalam bentuk tabel
Kondisi Akhir	Informasi harga terkini di tampilkan dalam bentuk tabel

17 **Skenario *Use Case* Mengirimkan Obrolan**

Tabel 4.56 Skenario *Use Case* Mengirimkan Obrolan

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_17

Nama	Mengirimkan obrolan
Tujuan	Untuk dapat mengirim obrolan kepada konsumen
Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat mengirim obrolan kepada konsumen.
Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah dalam keadaan login dan terkoneksi dengan jaringan internet. Penjual dalam halaman detail pesanan.
Aksi Aktor	Reaksi Sistem
1. Penjual memilih tombol kirim pesan	2. Sistem menampilkan halaman obrolan berupa text input
3. Penjual memilih tombol kirim	4. Sistem menerima input dan mengirim obrolan kepada konsumen.
Skenario alternatif 1: Jika penjual sudah pernah mengirim obrolan kepada konsumen.	
5. Penjual memilih menu obrolan pada drawer	6. Sistem menampilkan daftar obrolan pada halaman obrolan.
7. Penjual memilih konsumen yang ingin dikirim obrolan	8. Sistem menampilkan halaman obrolan berupa text input.
9. Penjual memilih tombol kirim	10. Sistem menerima input dan mengirim obrolan kepada penjual.
Kondisi Akhir	Obrolan terkirim kepada konsumen.

18 Skenario *Use Case* Mengelola Lokasi

Tabel 4.57 Skenario *Use Case* Mengelola Lokasi

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_18
Nama	Mengelola lokasi
Tujuan	Untuk dapat mengelola lokasi berjualan

Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat mengelola lokasi berjualan berikut waktu berjualan.
Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah dalam keadaan login dan terkoneksi dengan jaringan internet. Penjual berada pada halaman pengaturan
Aksi Aktor	Reaksi Sistem
1. Penjual memilih menu atur lokasi	2. Sistem menampilkan detail pengaturan lokasi berupa dropdown area berjualan, textinput waktu berjualan dan kolom berbagi lokasi berupa switch panel.
3. Penjual merubah data yang dinginkan	
4. Penjual memilih menu simpan	5. Sistem menerima data input penjual dan menyimpan kedalam *database*
Kondisi Akhir	Data lokasi penjual berhasil diperbarui.

19 Skenario *Use Case* Logout

Tabel 4.58 Skenario *Use Case* Logout

Skenario Kasus pada Sistem	
Nomor *Use Case*	MLJ_003_19
Nama	Logout
Tujuan	Untuk dapat keluar dari sistem
Deskripsi	*Use Case* ini menjelaskan bagaimana penjual dapat melakukan logout untuk keluar dari sistem.
Aktor	Penjual
Skenario Utama	
Kondisi Awal	Penjual telah terkoneksi dengan jaringan internet dan dalam keadaan session login
Aksi Aktor	Reaksi Sistem

1. Penjual memilih menu drawer	2. Sistem menampilkan halaman drawer
3. Penjual memilih menu logout	4. Sistem menerima input dan menghapus session login.
Kondisi Akhir	Sistem menampilkan halaman login.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

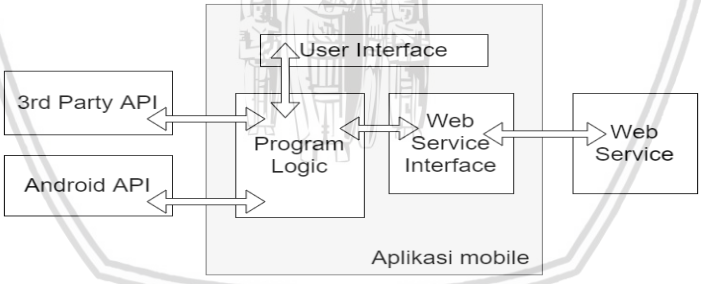
Bab ini menjelaskan fase perancangan dan implementasi dari pengembangan sistem katalog dan pemesanan produk kebutuhan dapur berbasis android, *MLIJO*. Pada tahapan ini terdapat beberapa tahap yang akan mengadopsi metode dari *Mobile-D,* yaitu fase *initialize* pada perancangan dan fase *productionize* serta *stabilize* pada implementasi sistem.

5.1 Perancangan Sistem

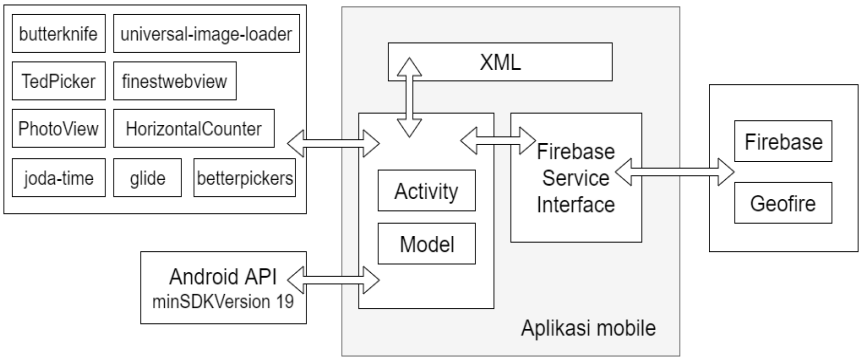
Perancangan sistem dilakukan setelah proses analisis kebutuhan dilakukan. Tahapan perancangan ini juga mengadopsi beberapa tahapan yang ada dalam metode pengembangan *Mobile-D* yang terdiri dari beberapa tahapan yaitu, perancangan arsitektur sistem, perancangan basisdata, perancangan diagram *class,* perancangan diagram *sequence,* perancangan algoritme dan perancangan antarmuka pengguna.

5.1.1 Perancangan Arsitektural

Sistem yang dibangun pada penelitian ini menggunakan arsitektur tiga *layer.* Dimana ketiga *layer* tersebut terdiri dari *layer* API pihak ketiga, *layer* aplikasi android itu sendiri dan *layer web service*. Ketiga *layer* tersebut harus dapat berkomunikasi dengan baik untuk memberikan fungsionalitas yang diperlukan. Berikut ini merupakan rancangan arsitektur sistem katalog dan pemesanan produk kebutuhan dapur berbasis android, *MLIJO* seperti pada gambar 5.1 dan 5.2:



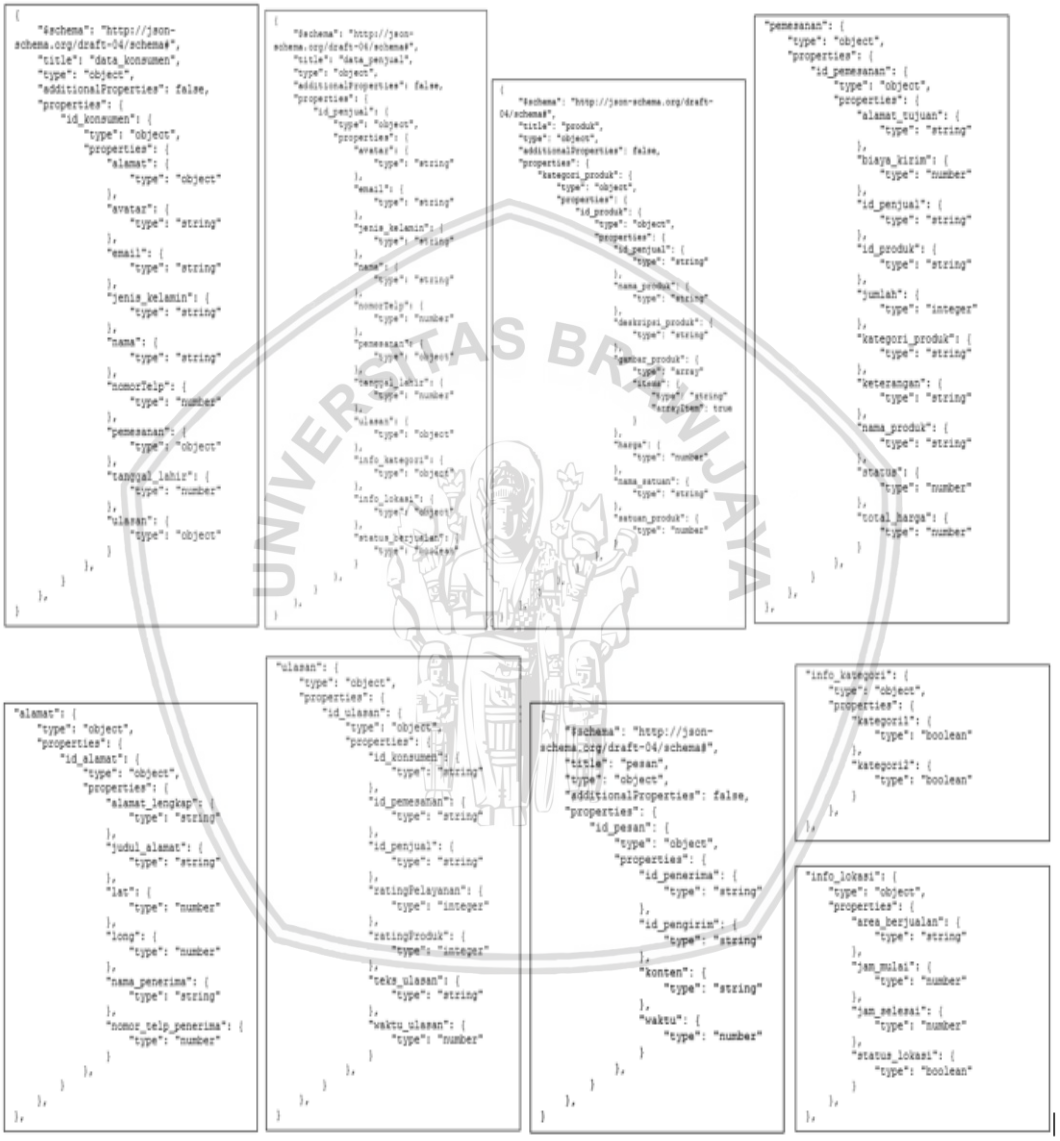
Gambar 5.1 Arsitektur Sistem Aplikasi *Mobile*
Sumber : (STAPIĆ, 2013)



Gambar 5.2 Arsitektur Sistem Aplikasi MLIJO

5.1.2 Perancangan Basis Data

Perancangan basis data pada sistem ini dilakukan dengan skema diagram karena pada pengembangan aplikasi ini menggunakan *database Firebase* yang berbasis noSQL. Pada *database* tersebut juga tidak terdapat relasi sebagaimana noSQL yang tidak menerapkan relasi antar diagram. Gambar *database schema* di tunjukan pada gambar 5.3 berikut, serta dijelaskan lebih lanjut pada tabel 5.1 hingga 5.11 untuk perancangan *database* pada sistem ini :



Gambar 5.3 Database *schema* Aplikasi MLIJO

1. Tabel Konsumen

   Nama tabel : data_konsumen
   Jumlah field : 6
   Fungsi : untuk menyimpan data konsumen

Tabel 5.1 Struktur tabel Konsumen

No	Nama *Field*	Tipe	Deskripsi
1	Id_konsumen	document	Id Konsumen
2	daftarTransaksi	document	Daftar transaksi konsumen
3	detailKonsumen	document	Detail informasi konsumen
4	deviceToken	String	Id perangkat konsumen
5	obrolan	document	Daftar obrolan konsumen
6	uid	String	Id Konsumen

2. Tabel Detail Konsumen
   Nama tabel : detailKonsumen
   Jumlah field : 7
   Fungsi : untuk menyimpan data detail konsumen

Tabel 5.2 Struktur tabel Detail Konsumen

No	Nama *Field*	Tipe	Deskripsi
1	NIK	Integer	Nomor identitas konsumen
2	alamat	String	Alamat konsumen
3	avatar	String	Foto profil konsumen
4	latitude	double	Koordinat latitude konsumen
5	Longitude	double	Koordinat longitude konsumen
6	nama	String	Nama lengkap konsumen
7	noTelp	Integer	Nomor telepon konsumen

3. Tabel Penjual
   Nama tabel : data_penjual
   Jumlah field : 10
   Fungsi : untuk menyimpan data penjual

Tabel 5.3 Struktur tabel Penjual

No	Nama *Field*	Tipe	Deskripsi
1	Id_penjual	document	Id penjual
2	daftarTransaksi	document	Daftar transaksi penjual
3	detailPenjual	document	Detail informasi penjual
4	deviceToken	String	Id perangkat penjual
5	infoKategori	document	Daftar ketegori produk penjual

6	infoLokasi	document	Detail informasi lokasi penjual
7	obrolan	document	Daftar obrolan penjual
8	statusBerjualan	Boolean	Status aktivasi akun penjual
9	statusLokasi	Boolean	Status berbagi lokasi penjual
10	Uid	String	Id penjual

4. Tabel detail penjual
   Nama tabel : detailPenjual
   Jumlah field : 5
   Fungsi : untuk menyimpan data detail penjual

Tabel 5.4 Struktur tabel Detail Penjual

No	Nama *Field*	Tipe	Deskripsi
1	NIK	Integer	Nomor identitas penjual
2	alamat	String	Alamat penjual
3	avatar	String	Foto profil penjual
4	nama	String	Nama lengkap penjual
5	noTelp	Integer	Nomor telepon penjual

5. Tabel Lokasi
   Nama tabel : infoLokasi
   Jumlah field : 5
   Fungsi : untuk menyimpan data informasi lokasi berjualan

Tabel 5.5 Struktur tabel Lokasi

No	Nama *Field*	Tipe	Deskripsi
1	hariMulai	String	Informasi hari berjualan
2	hariSelesai	String	Informasi hari berjualan
3	jamMulai	String	Informasi jam berjualan
4	jamSelesai	String	Informasi jam bejualan
5	kecamatan	String	Informasi area lokasi berjualan

6. Tabel Geofire
   Nama tabel : geofire
   Jumlah field : 3
   Fungsi : untuk menyimpan data geofire penjual

Tabel 5.6 Struktur tabel Geofire

No	Nama *Field*	Tipe	Deskripsi
1	Id_penjual	document	Id Penjual
2	g	String	Id Geohash
3	l	array	Latitude dan longitude

7. Tabel Transaksi
   Nama tabel : daftarTransaksi
   Jumlah field : 15
   Fungsi : untuk menyimpan informasi daftar transaksi

Tabel 5.7 Struktur tabel Transaksi

No	Nama *Field*	Tipe	Deskripsi
1	Id_transaksi	document	Id transaksi
2	biayaKirim	double	Nominal biaya kirim
3	catatanKonsumen	String	Pesan untuk penjual
4	idKonsumen	String	Id konsumen
5	idPenjual	String	Id penjual
6	idProduk	String	Id produk
7	idTransaksi	String	Id transaksi
8	jenisProduk	String	Tipe produk
9	jumlahOrderProduk	Integer	Jumlah pembelian produk
10	namaPenerima	String	Nama penerima
11	statusTransaksi	Integer	Status transaksi
12	tanggalKirim	String	Informasi tanggal pengiriman
13	tanggalPesan	long	Informasi tanggal pemesanan
14	totalHarga	double	Informasi total harga produk
15	waktuKirim	String	Informasi jam pengiriman

8. Tabel Kategori
   Nama tabel : infoKategori
   Jumlah field : 8
   Fungsi : untuk menyimpan informasi kategori produk

Tabel 5.8 Struktur tabel Kategori

No	Nama *Field*	Tipe	Deskripsi
1	kategoriAlat	boolean	Informasi kategori alat – alat
2	kategoriBuah	boolean	Informasi kategori buah
3	kategoriBumbu	boolean	Informasi kategori bumbu dapur
4	kategoriDaging	boolean	Informasi kategori daging
5	kategoriIkan	boolean	Informasi kategori makanan laut
6	kategoriLain	boolean	Informasi kategori lainnya
7	kategoriPalawija	boolean	Informasi kategori palawija
8	kategoriSayuran	boolean	Informasi kategori sayuran

9. Tabel Obrolan

   Nama tabel : obrolan

   Jumlah field : 9

   Fungsi : untuk menyimpan data obrolan pengguna

Tabel 5.9 Struktur tabel Obrolan

No	Nama *Field*	Tipe	Deskripsi
1	Id_pengirim	document	Id pengirim obrolan
2	Id_penerima	document	Id penerima obrolan
3	Id_obrolan	document	Id obrolan
4	idPenerima	String	Id penerima obrolan
5	idPengirim	String	Id pengirim obrolan
6	konten	String	Isi obrolan
7	kontenFoto	boolean	Informasi isi konten
8	kontenPengirim	boolean	Informasi konten milik pengirim
9	timestamp	long	Waktu kirim obrolan

10. Tabel Produk reguler

    Nama tabel : produk_reguler

    Jumlah field : 12

    Fungsi : untuk menyimpan informasi data produk reguler

Tabel 5.10 Struktur tabel Produk Reguler

No	Nama *Field*	Tipe	Deskripsi

1	Id_produk	document	Id produk
2	deskripsiProduk	String	Deskripsi produk
3	digitSatuan	Integer	Nominal satuan produk
4	gambarProduk	Array	Gambar produk
5	hargaProduk	double	Harga produk
6	idKategori	String	Id kategori produk
7	idLokasi	String	Id lokasi produk dijual
8	idPenjual	String	Id penjual
9	idProduk	String	Id produk
10	namaProduk	String	Nama produk
11	namaSatuan	String	Besaran satuan produk
12	waktuDibuat	long	Waktu pembaruan produk

11. Tabel Produk Khusus

Nama tabel : produk_khusus

Jumlah field : 5

Fungsi : untuk menyimpan informasi data produk khusus

Tabel 5.11 Struktur tabel Produk Khusus

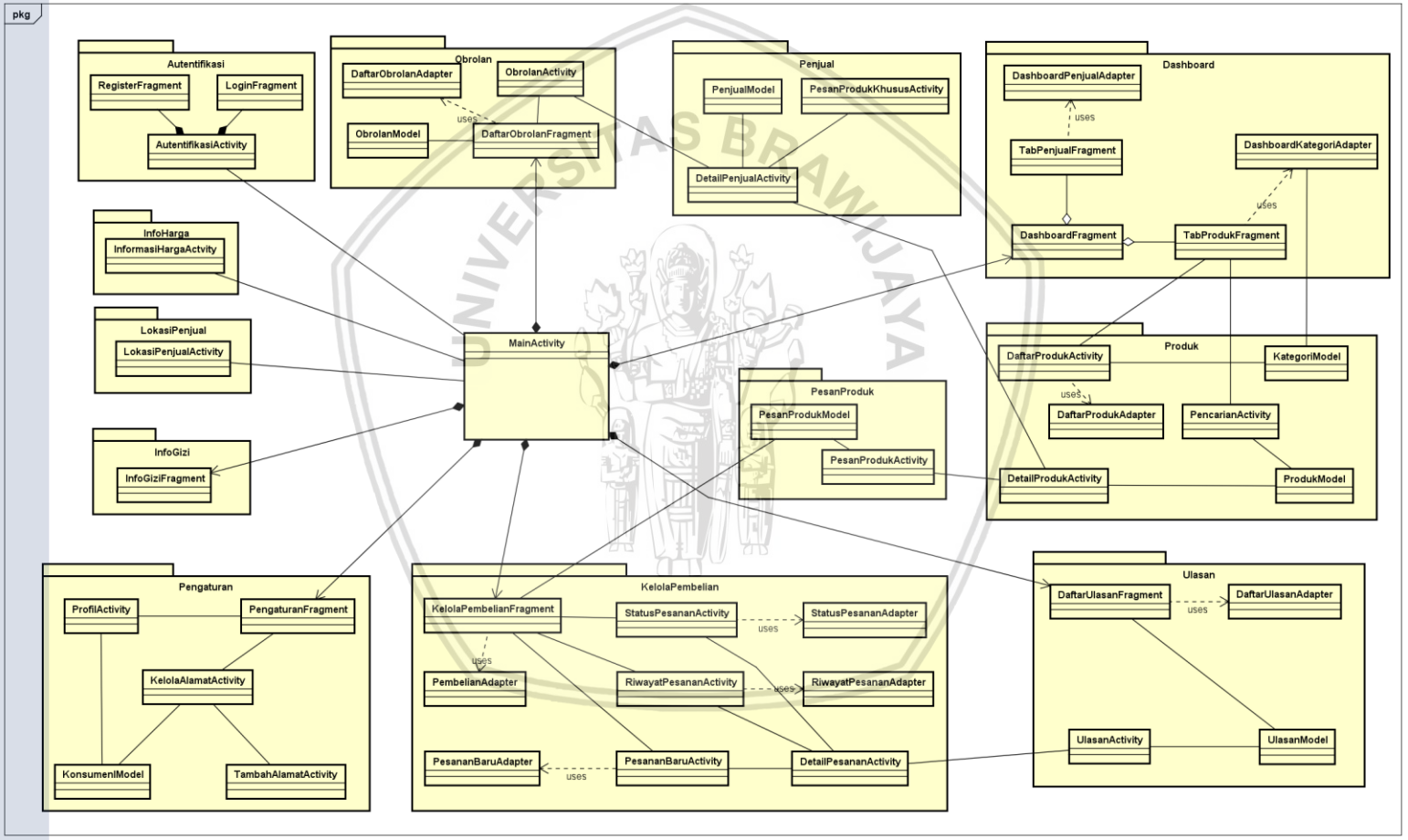
No	Nama *Field*	Tipe	Deskripsi
1	Id_produk	document	Id produk
2	digitSatuan	Integer	Nominal satuan produk
3	idProduk	String	Id produk
4	namaProduk	String	Nama produk
5	namaSatuan	String	Besaran satuan produk

5.1.3 Perancangan Diagram *Class*

Perancangan diagram *class* merupakan diagram yang digunakan untuk menujukkan kelas dan keterhubungannya di dalam suatu sistem. Pada sistem ini *class* di kelompokan dalam paket berdasarkan fungsi - fungsi yang akan diimplementasikan ke dalam sistem. Pada penelitian ini terbagi menjadi dua sistem yaitu konsumen dan penjual.

5.1.3.1 Diagram *Class* Konsumen

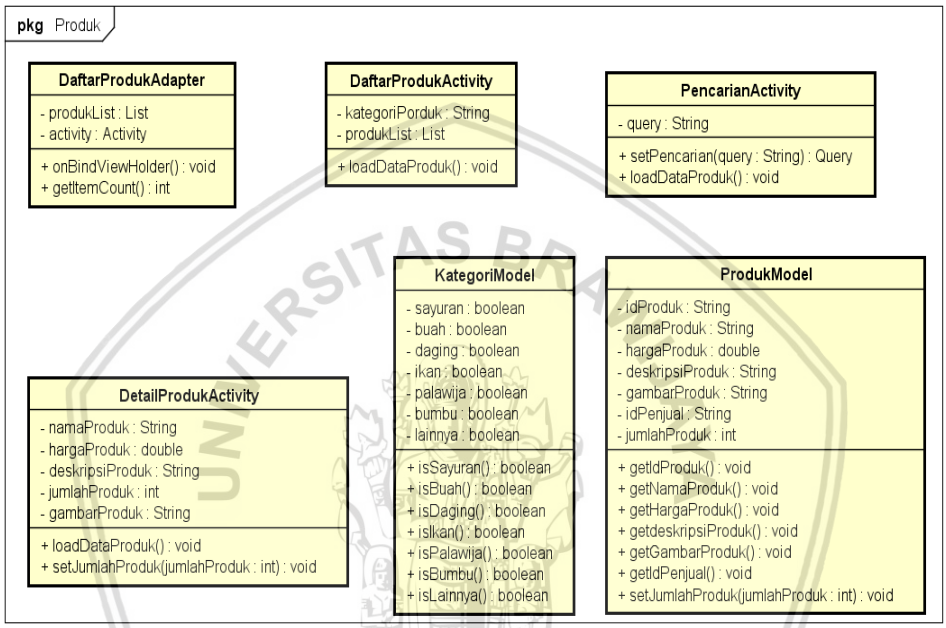
Pada bagian pertama menunjukan *class* diagram dari konsumen beserta relasi antar *class*. Berikut diagram *class* konsumen seperti pada gambar 5.4 :



Gambar 5.4 Diagram *Class* Konsumen

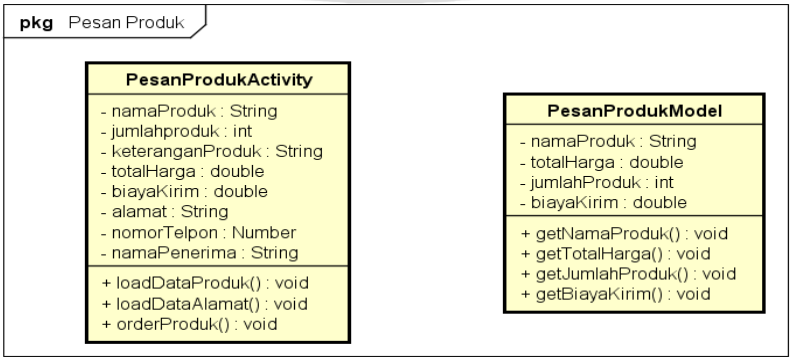
Dari keseluruhan diagram *class* konsumen pada gambar 5.4, diambil tiga paket utama untuk diperjelas atribut dan operasi pada setiap *class*. Ketiga paket utama tersebut yaitu, paket `Produk`, paket `PesanProduk` dan paket `Penjual`.

Paket `Produk` merupakan paket yang menjelaskan tentang *class* yang berkaitan dengan produk, seperti daftar produk, detail produk dan lainnya. Paket ini terdiri dari enam *class*, yaitu *class* `DaftarProdukActivity`, `DetailProdukActivity`, `PencarianActivity`, `DaftarProdukAdapter`, `KategoriModel`, dan `ProdukModel`. Atribut dan operasi dari *class* tersebut dapat dilihat pada gambar 5.5 :



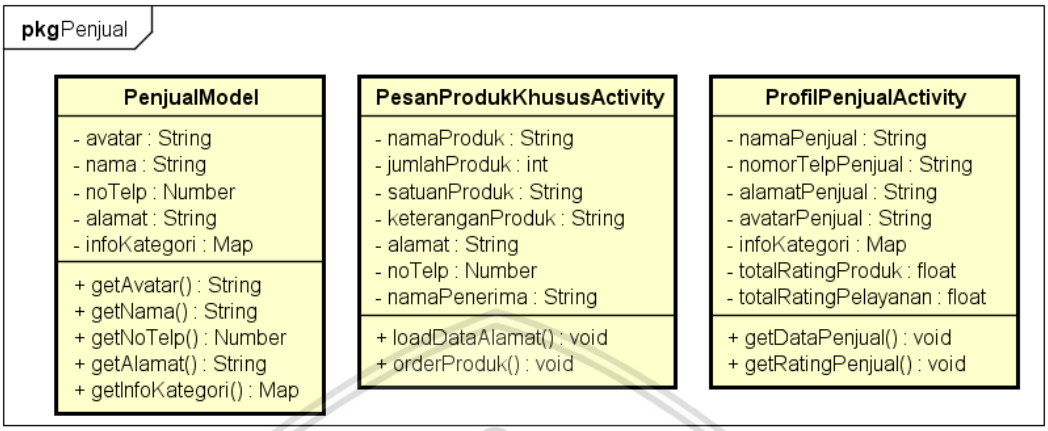
Gambar 5.5 Diagram *Class* Paket Produk

Paket pesan produk merupakan paket yang menjelaskan tentang *class* yang berkaitan dengan pemesanan produk, seperti `PesanProdukActivity` dan `PesanProdukModel`. Atribut dan operasi dari *class* tersebut dapat dilihat pada gambar 5.6 berikut :



Gambar 5.6 Diagram *Class* Paket Pesan Produk

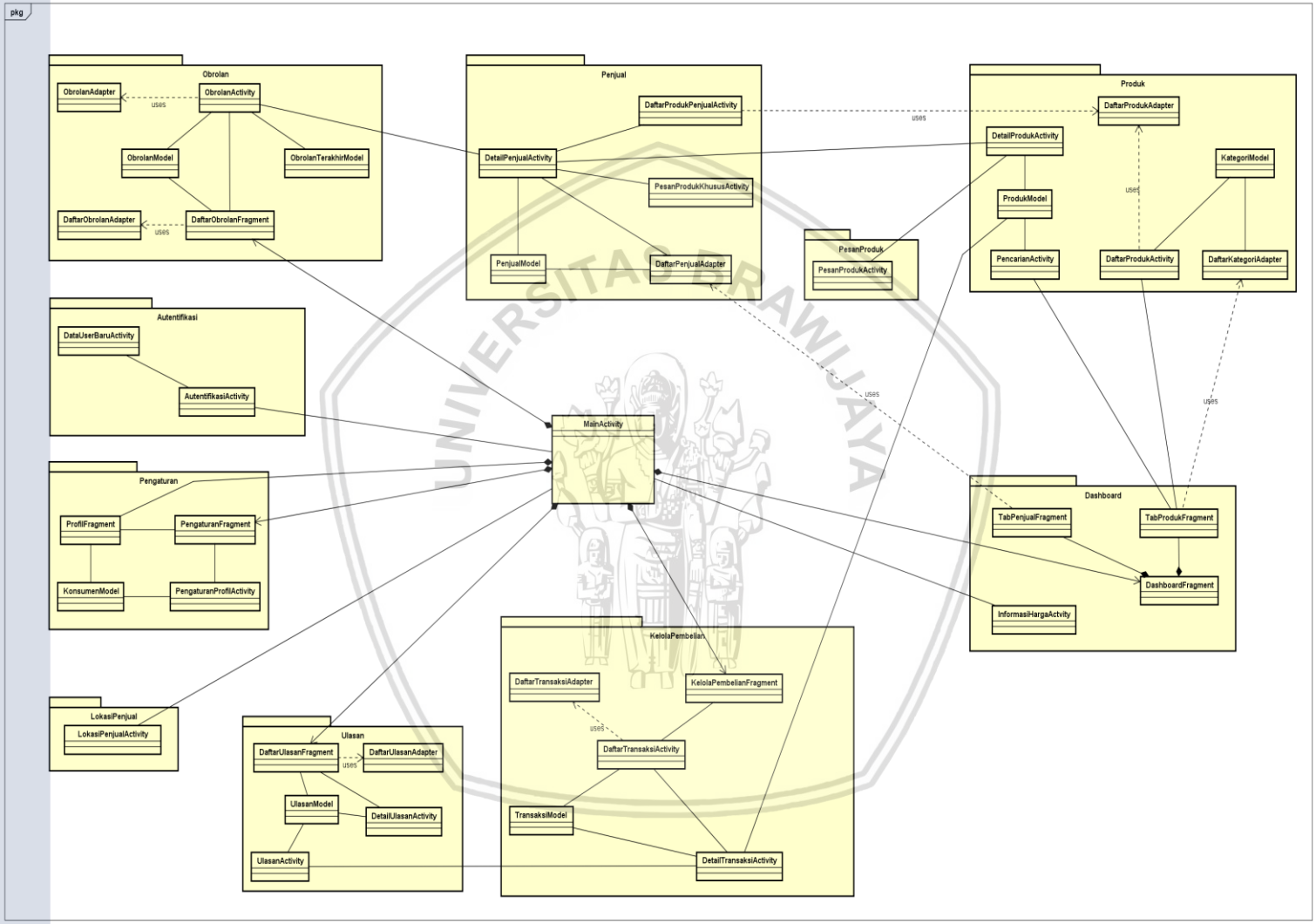
Paket penjual merupakan paket yang menjelaskan tentang *class* yang berkaitan dengan *class* `PenjualModel`, `ProfilPenjualActivity` dan `PesanProdukKhususActivity`. Atribut dan operasi dari *class* tersebut dapat dilihat pada gambar 5.7 berikut :



Gambar 5.7 Diagram *Class* Paket Penjual

5.1.3.2 Diagram Class Konsumen Iterasi 0

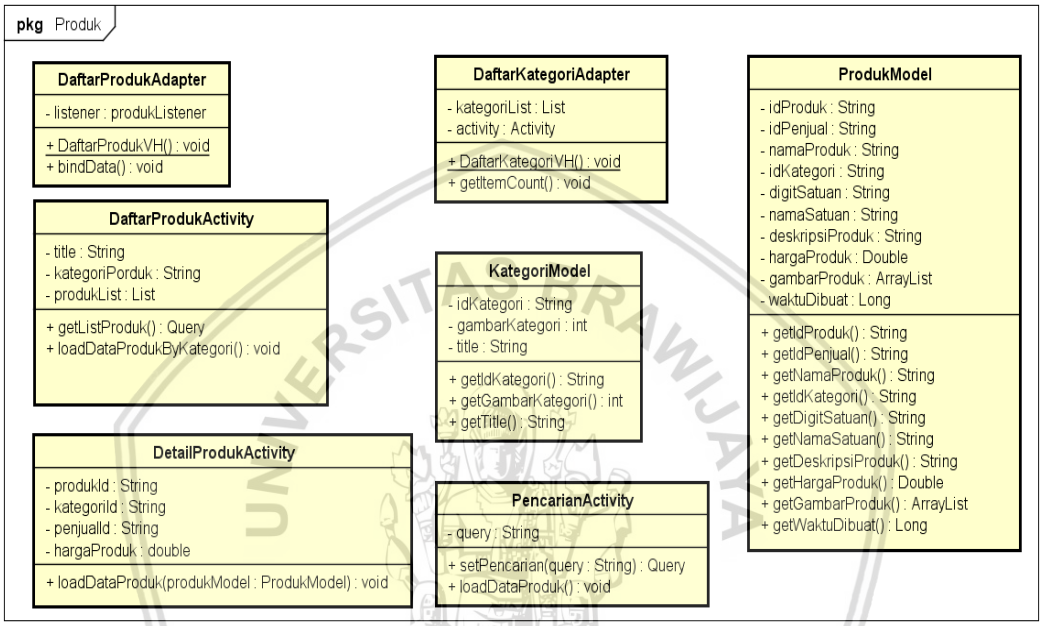
Diagram *class* konsumen iterasi 0 merupakan perubahan *class* diagram hasil dari *feedback* yang di dapatkan dari pengguna setelah fase *initialize*. Perubahan ini dilakukan untuk memenuhi kebutuhan sesuai dengan keinginan pengguna. Selain hasil *feedback* dari pengguna, perubahan pada iterasi 0 juga berasal dari pengembang itu sendiri. Perubahan dilakukan untuk beberapa alasan, seperti menambahkan fungsi yang belum ada sebelumnya, merubah fungsi yang sudah ada, menyederhanakan beberapa *class* maupun melakukan revisi penamaan *class*. Terdapat banyak perubahan pada iterasi 0 ini, diantaranya pada paket `Dashboard`, `Produk`, `PesanProduk`, `Ulasan`, `Penjual`, `Obrolan` `KelolaPembelian`, `Pengaturan`, dan paket `Autentifikasi`. Sementara paket `InfoGizi` dihapus karena pengguna memberikan penilaian di bawah 4 terhadap fungsi tersebut dan paket `InfoHarga` dipindah *class*nya ke dalam paket `Dashboard`. Berikut diagram *class* konsumen hasil dari iterasi 0 beserta relasi antar *class*nya seperti pada gambar 5.8 :



Gambar 5.8 Diagram *Class* Konsumen Iterasi 0

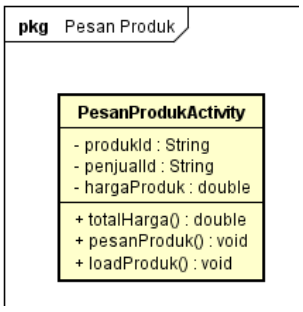
Dari keseluruhan diagram class konsumen hasil iterasi 0 pada gambar 5.8, kemudian hanya diambil tiga paket utama untuk diperjelas atribut dan operasi pada setiap class. Ketiga paket utama tersebut yaitu, paket `Produk`, paket `PesanProduk` dan paket `Penjual`. Berikut diagram class konsumen hasil dari iterasi 0 :

Pada paket produk hasil iterasi 0 mengalami perubahan berupa penambahan *class* `DaftarKategoriAdapter`, juga perubahan atribut pada beberapa *class*. Atribut dan operasi dari *class* tersebut dapat dilihat pada gambar 5.9 :



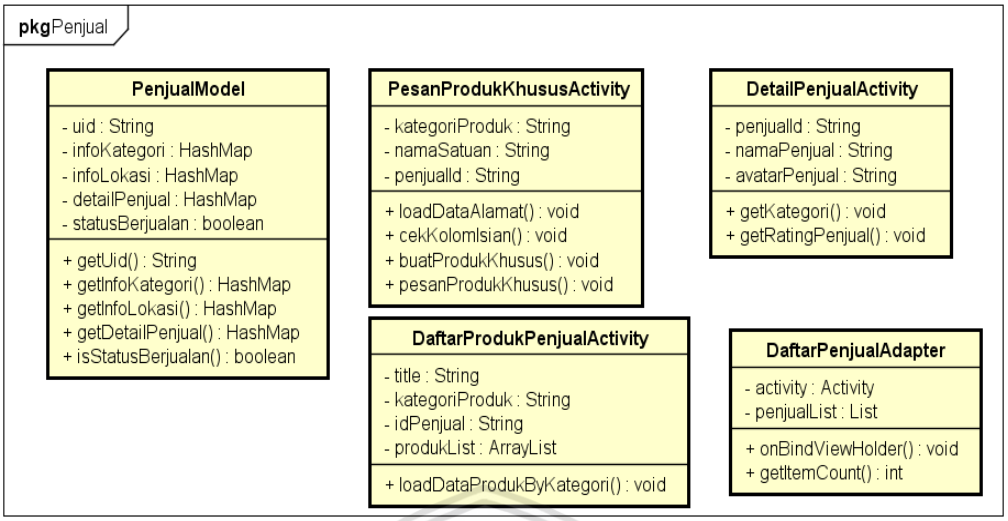
Gambar 5.9 Diagram *Class* Paket Produk Iterasi 0

Pada paket PesanProduk mengalami perubahan berupa penghapusan *class* `PesanProdukModel` dikarenakan pada proses pemesanan produk hanya dilakukan pada *class* `PesanProdukActivity` tanpa memerlukan fungsi dari *class* lainnya. Atribut dan operasi dari *class* tersebut dapat dilihat pada gambar 5.11 :



Gambar 5.10 Diagram *Class* Paket Penjual Iterasi 0

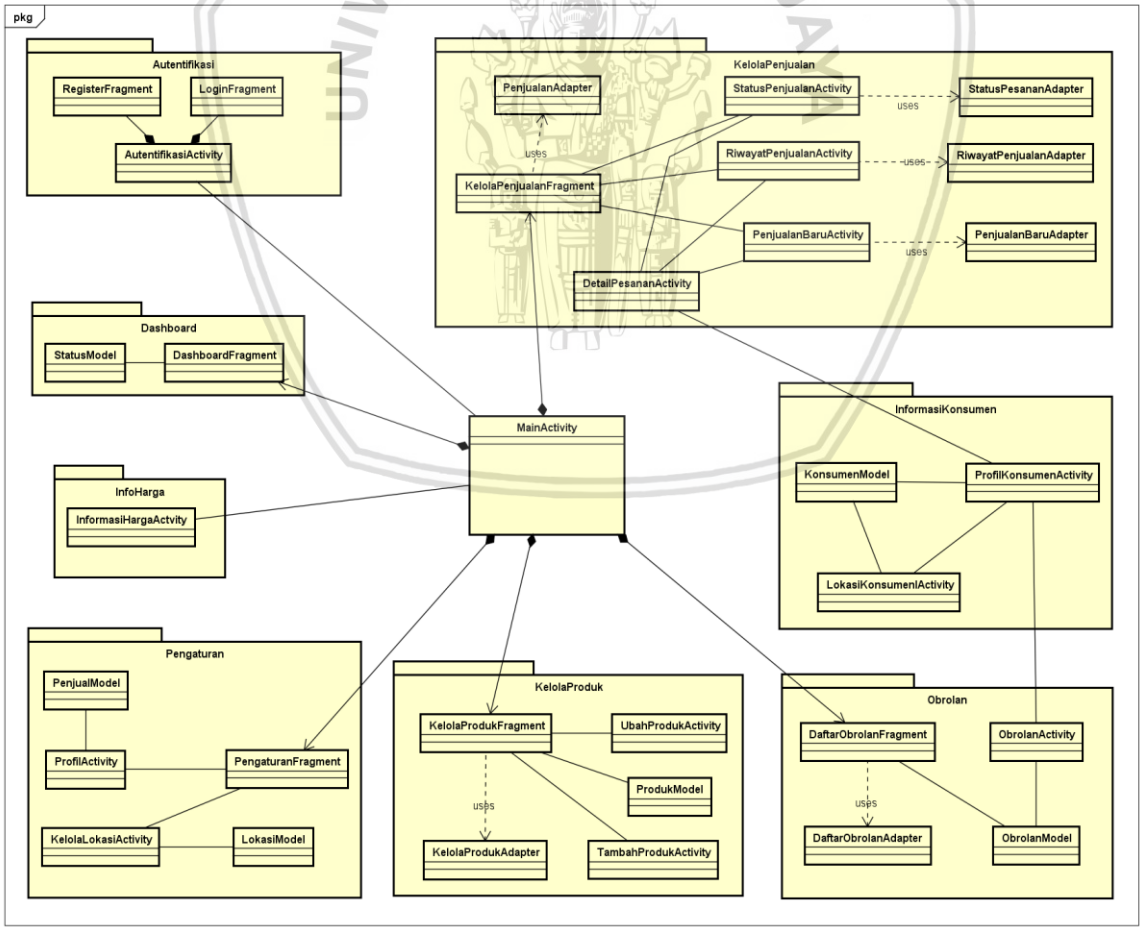
Pada paket `Penjual` mengalami perubahan berupa penambahan *class* `DaftarPenjualAdapter` dan *class* `DaftarProdukPenjualActivity`. Atribut dan operasi dari *class* tersebut dapat dilihat pada gambar 5.10 :



Gambar 5.11 Diagram *Class* Paket PesanProduk Iterasi 0

5.1.3.3 Diagram Class Penjual

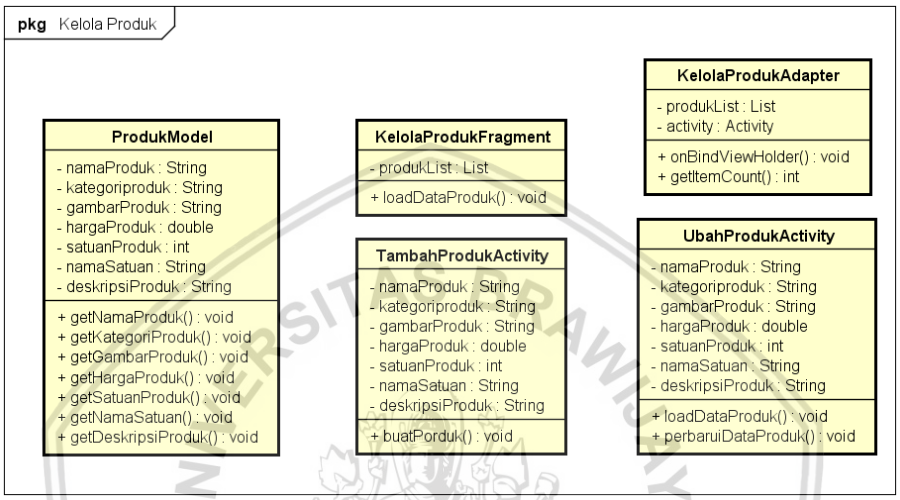
Pada bagian kedua menunjukan *class* diagram dari penjual beserta relasi antar *class*. Berikut diagram *class* penjual seperti pada gambar 5.12 :



Gambar 5.12 Diagram *Class* Penjual

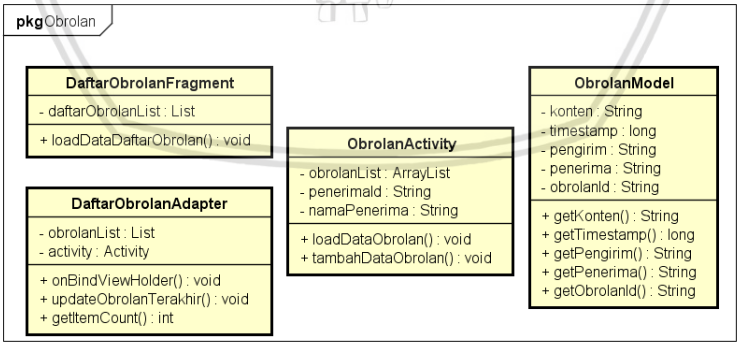
Dari keseluruhan diagram *class* penjual pada gambar 5.12, diambil tiga paket utama untuk diperjelas atribut dan operasi pada setiap *class*. Ketiga paket utama tersebut yaitu, `KelolaProduk`, `KelolaPenjualan` dan `Obrolan`.

Paket `KelolaProduk` merupakan paket yang menjelaskan tentang *class* yang berkaitan dengan *class* `KelolaProdukFragment`, `ProdukModel`, `KelolaProdukAdapter`, `BuatProdukActivity`, dan `UbahProdukActivity`. Atribut dan operasi dari *class* tersebut dapat dilihat pada gambar 5.13 berikut:



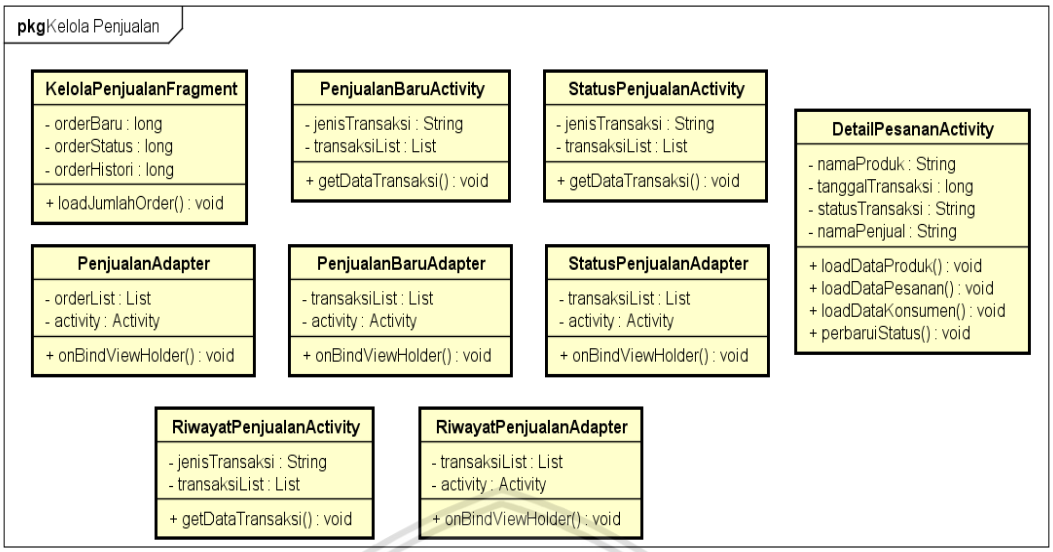
Gambar 5.13 Diagram *Class* Paket Kelola Produk

Paket `Obrolan` merupakan paket yang menjelaskan tentang *class* yang berkaitan dengan *class* `DaftarObrolanFragment`, `DaftarObrolanAdapter`, `ObrolanActivity`, dan `ObrolanModel`. Atribut dan operasi dari *class* tersebut dapat di lihat pada gambar 5.15 berikut:



Gambar 5.14 Diagram *Class* Paket Obrolan

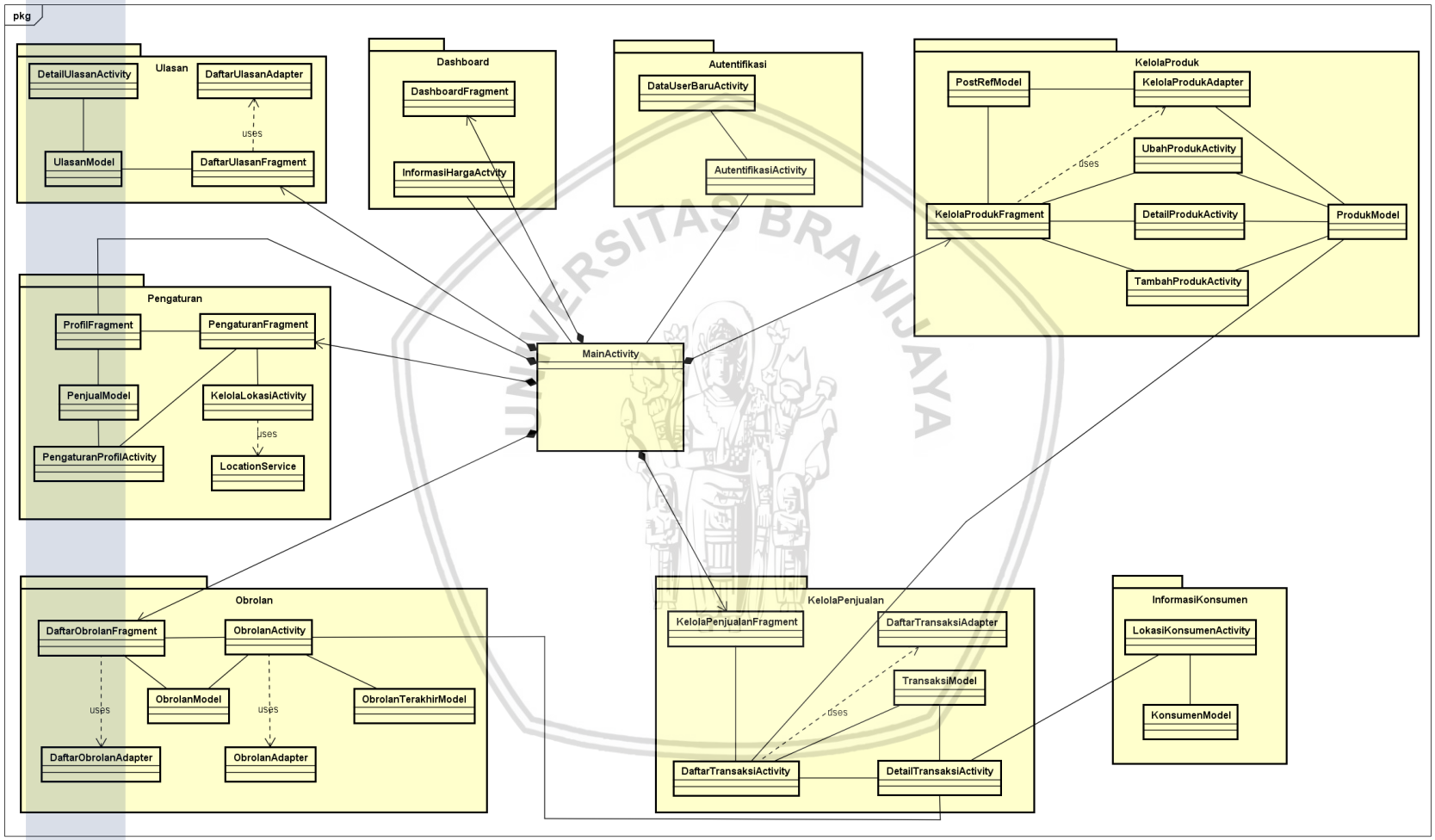
Paket `KelolaPenjualan` merupakan paket yang menjelaskan tentang *class* yang berkaitan dengan *class* `KelolaPenjualanFragment`, `PenjualanAdapter`, `StatusPenjualanActivity`, `StatusPenjualanAdapter`, `PenjualanBaruActivity`, `PenjualanBaruAdapter`, `RiwayatPenjualanActivity`, `RiwayatPenjualanAdapter` dan `DetailPesananActivity`. Atribut dan operasi dari *class* tersebut dapat di lihat pada gambar 5.14 berikut:



Gambar 5.15 Diagram *Class* Paket Kelola Penjualan

5.1.3.4 Diagram Class Penjual Iterasi 0

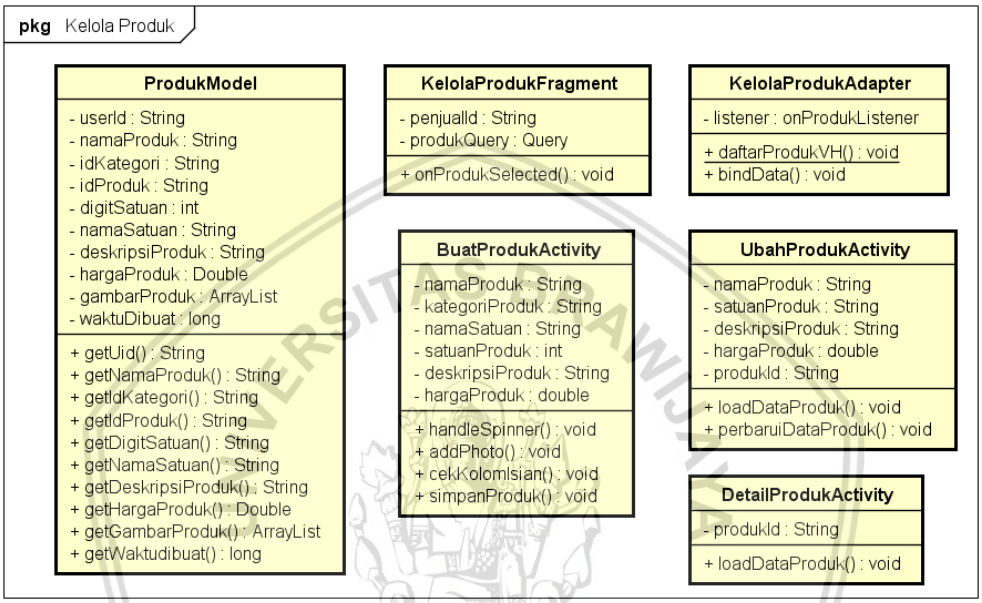
Pada diagram *class* penjual iterasi 0 merupakan perubahan *class* diagram hasil dari *feedback* yang di dapatkan dari pengguna setelah fase *initialize*. Perubahan ini dilakukan untuk memenuhi kebutuhan sesuai dengan keinginan pengguna. Selain hasil *feedback* dari pengguna, perubahan pada iterasi 0 juga berasal dari pengembang itu sendiri. Perubahan dilakukan untuk beberapa alasan, seperti menambahkan fungsi yang belum ada sebelumnya, merubah fungsi yang sudah ada, menyederhanakan beberapa *class* maupun melakukan revisi penamaan *class*. Terdapat banyak perubahan pada iterasi 0 ini, diantaranya pada paket `Dashboard`, `Autentifikasi`, `KelolaProduk`, `KelolaPenjualan`, `InformasiKonsumen`, `Obrolan` dan paket `Pengaturan`. Sementara paket `InfoHarga` dihapus dan *class* `InformasiHargaActivity` dipindah ke dalam paket `Dashboard`. Kemudian juga terdapat penambahan paket `Ulasan`. Berikut diagram *class* Penjual hasil dari iterasi 0 beserta relasi antar *class*nya seperti pada gambar 5.16 :



Gambar 5.16 Diagram *Class* Penjual Iterasi 0

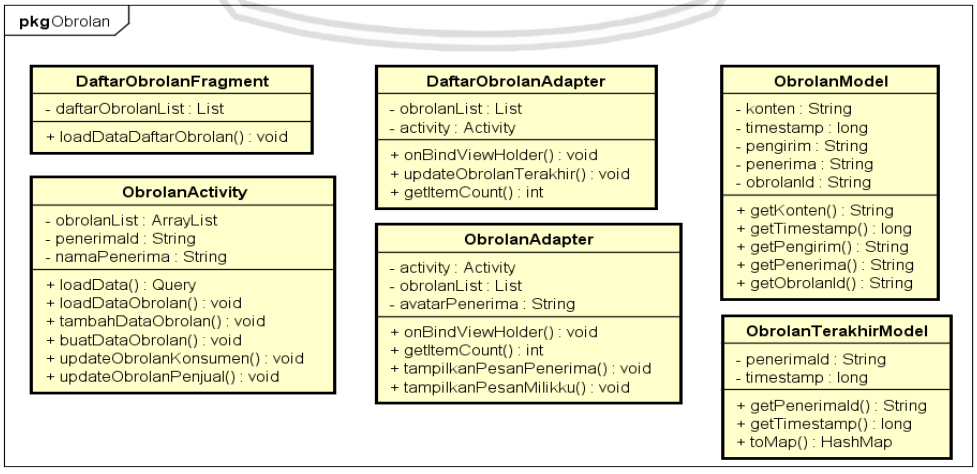
Dari keseluruhan diagram *class* penjual hasil iterasi 0 pada gambar 5.16, kemudian hanya diambil tiga paket utama untuk diperjelas atribut dan operasi pada setiap *class*. Ketiga paket utama tersebut yaitu, paket `KelolaProduk`, `KelolaPenjualan dan Obrolan`.

Pada paket KelolaProduk hasil iterasi 0 mengalami perubahan berupa penambahan *class* `DetailProdukActivity`, juga perubahan atribut pada beberapa *class*. Atribut dan operasi dari *class* tersebut dapat dilihat pada gambar 5.5 :



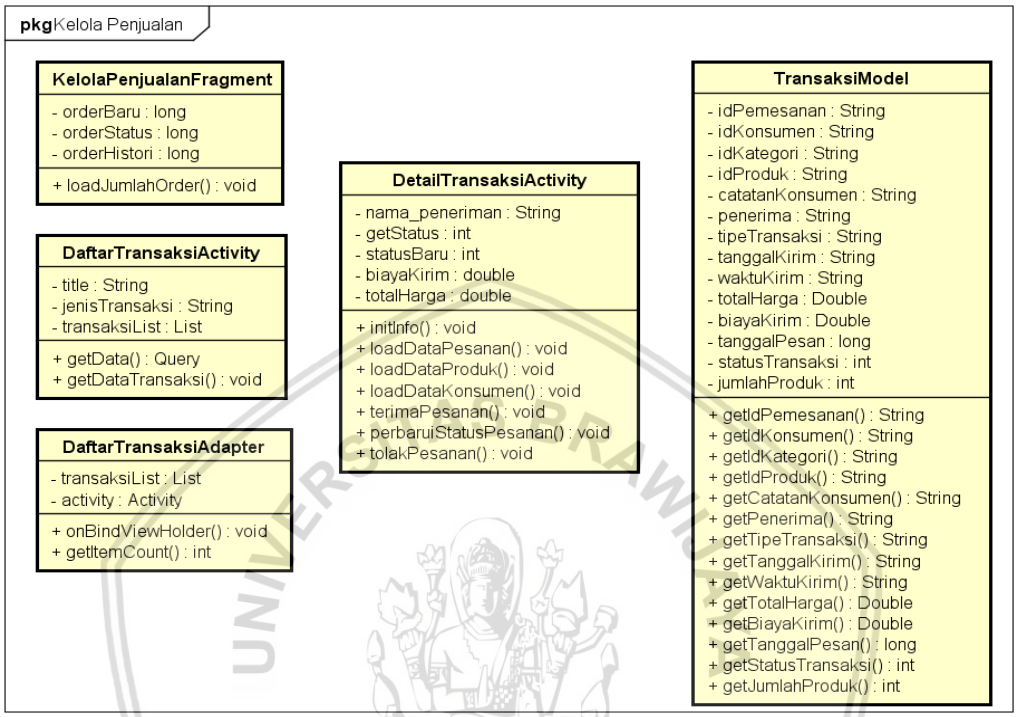
Gambar 5.17 Diagram *Class* Penjual Paket KelolaProduk Iterasi 0

Pada paket Obrolan hasil iterasi 0 mengalami perubahan berupa penambahan *class* `ObrolanAdapter` dan `ObrolanTerakhirModel`, juga perubahan atribut pada beberapa *class*. Atribut dan operasi dari *class* tersebut dapat dilihat pada gambar 5.5 :



Gambar 5.18 Diagram *Class* Penjual Paket Obrolan Iterasi 0

Pada paket KelolaPenjualan hasil iterasi 0 mengalami beberapa perubahan seperti yang telah di jelaskan pada awal dari sub-bab ini. Perubahan tersebut terjadi karena adanya perbaikan penulisan kode program sehingga membuat *class* menjadi lebih sederhana. Atribut dan operasi dari *class* tersebut dapat dilihat pada gambar 5.19:



Gambar 5.19 Diagram *Class* Penjual Paket KelolaPenjualan Iterasi 0

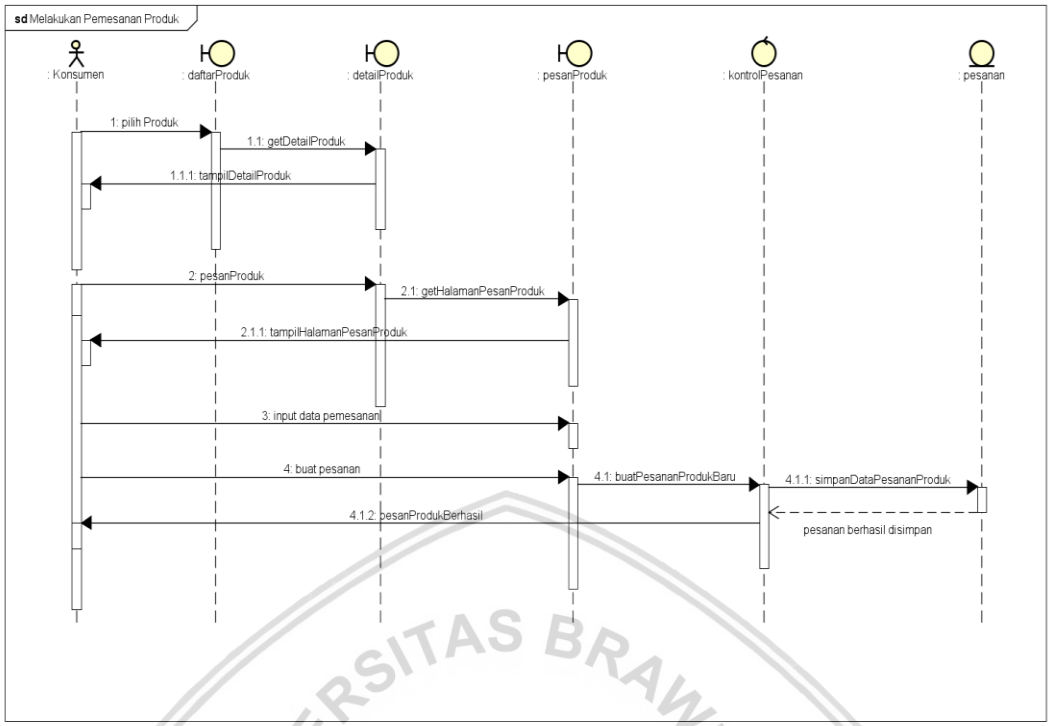
5.1.4 Perancangan Diagram Sequence

Diagram *sequence* merupakan representasi dari proses interaksi antar objek sesuai dengan urutan prosesnya. Diagram *Sequence* menunjukkan alur bagaimana pertukaran serangkaian pesan antar objek-objek yang melakukan suatu aksi tertentu. Diagram *Sequence* digunakan untuk menggambarkan arus pekerjaan, pesan yang disampaikan dan elemen-elemen yang bekerja sama didalamnya dari waktu ke waktu untuk mencapai suatu tujuan tertentu.

Pada sistem MLIJO ini terdapat dua sub-sistem yang kemudian pada bagian ini diwakili oleh empat sequence diagram saja, dimana 2 diagram dari sub-sistem konsumen dan 2 diagram dari sub-sistem penjual seperti pada gambar berikut :

a. Melakukan Pesan Produk (Konsumen)

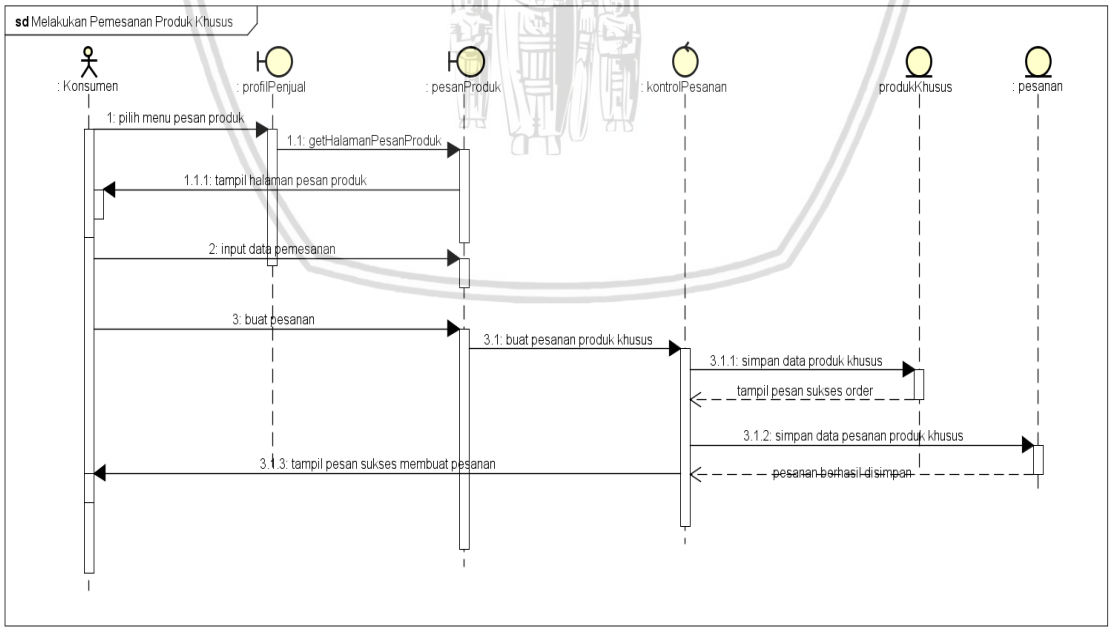
Pada sequence diagram berikut menjelaskan bagaimana alur pada sistem MLIJO saat Konsumen melakukan pemesanan produk yang telah tersedia pada daftar produk.



Gambar 5.20 *Sequence* Diagram Melakukan Pemesanan Produk

b. Melakukan Pemesanan Produk Khusus(Konsumen)

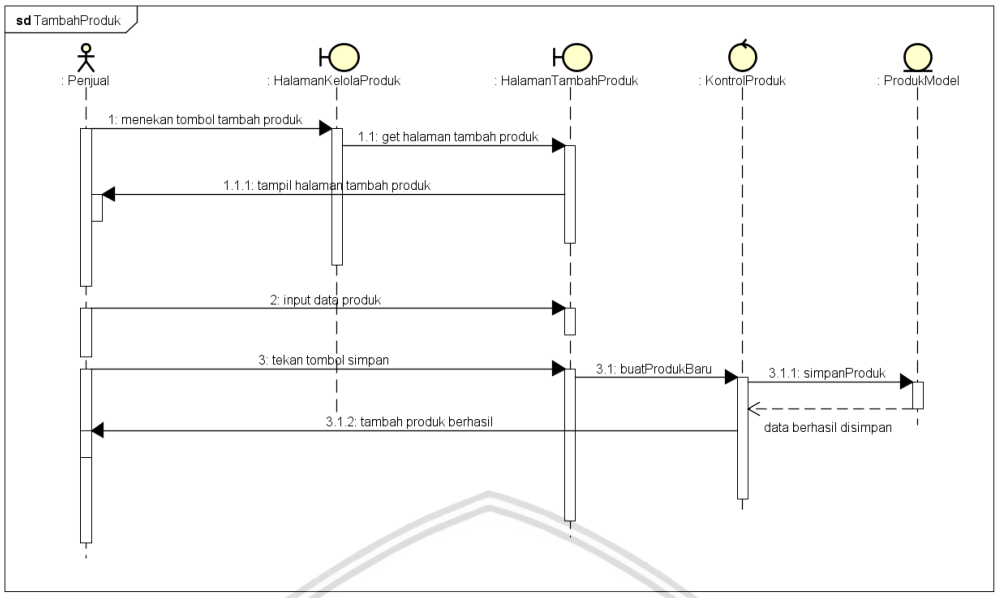
Pada sequence diagram berikut menjelaskan bagaimana alur pada sistem MLIJO saat konsumen melakukan pemesanan produk yang tidak tersedia di aplikasi kepada penjual tertentu.



Gambar 5.21 *Sequence* Diagram Melakukan Pemesanan Produk Khusus

c. Melakukan Tambah Produk (Penjual)

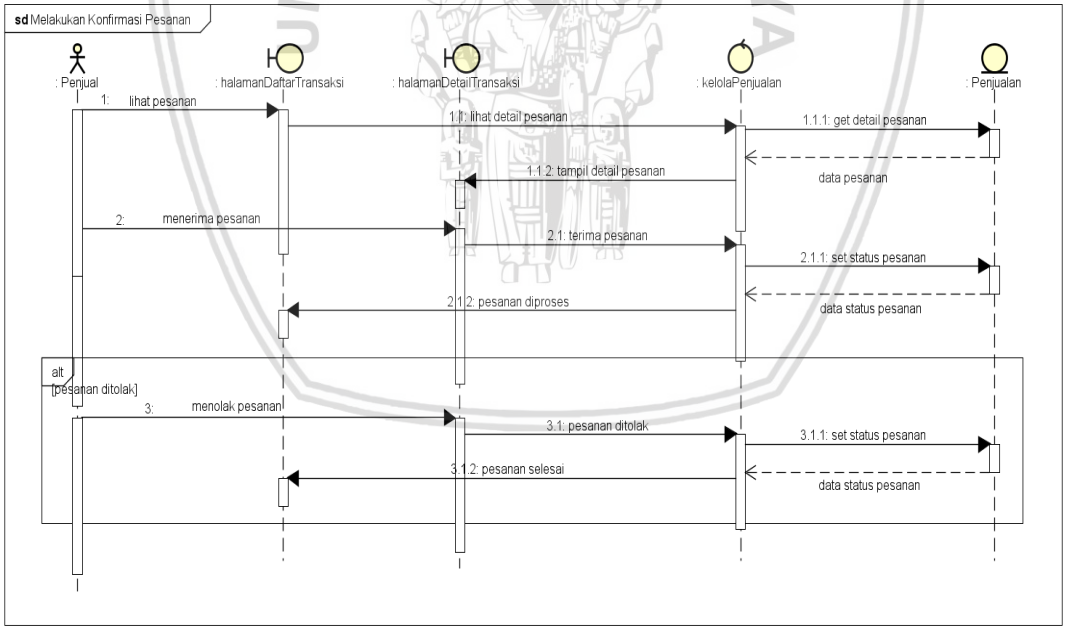
Pada sequence diagram berikut menjelaskan bagaimana alur pada sistem MLIJO saat penjual melakukan tambah produk baru.



Gambar 5.22 Sequence Diagram Tambah Produk

d. Melakukan Konfirmasi Pesanan (Penjual)

Pada sequence diagram berikut menjelaskan bagaimana alur pada sistem MLIJO saat penjual melakukan konfirmasi terhadap pesanan produk yang dilakukan oleh konsumen.



Gambar 5.23 Sequence Diagram Konfirmasi Pesanan

5.1.5 Perancangan Algoritme

Perancangan algoritme merupakan bagian perancangan yang berfungsi untuk menjelaskan langkah-langkah untuk melakukan suatu proses dari fungsi tertentu. Pada bagian ini akan dijelaskan beberapa perancangan algoritme dari fungsi cek kolom isian, simpan produk, buat produk baru, buat produk khusus, dan pesan produk khusus.

5.1.5.1 Algoritme Cek Kolom Isian

Algoritme `cek kolom isian` merupakan algoritme yang berguna untuk melakukan proses pengecekan kolom data input apakah sudah terisi dengan semestinya atau tidak. Implementasi algoritma untuk `cek kolom isian` dijelaskan pada Gambar 5.24.

```
Algoritme cekKolomIsian

Deklarasi :
   - Boolean → nilaiHasil
   - String → inputNamaProduk, inputHargaProduk,
     inputNominalSatuan, inputDeskripsiProduk

Deskripsi :
   - Masukkan : inputNamaProduk, inputHargaProduk,
     inputNominalSatuan, inputDeskripsiProduk
   - Proses :
     1. Deklarasi variabel nilaiHasil
     2. Melakukan cek kondisi dimana nilai variabel dari edit
        text kosong atau tidak
     3. Menyimpan nilai false apabila nilai salah satu
        variabel kosong dan nilai true jika tidak kosong pada
        variabel nilaiHasil
     4. Mendapatkan nilai kembalian dari nilaiHasil
   - Keluaran : nilai variabel nilaiHasil
```

Gambar 5.24 Algoritme Method Cek Kolom Isian

5.1.5.2 Algoritme Simpan Produk

Algoritme `simpan produk` merupakan algoritme yang berguna untuk melakukan proses mendapatkan nilai variabel produk dari `edit text` dan juga *file* `uri` dari foto produk yang akan di upload. Implementasi algoritme untuk `simpan produk` dijelaskan pada Gambar 5.25

```
Algoritme simpanProduk

Deklarasi :

   - String → namaProduk, hargaProduk, satuanProduk,
     deskripsiProduk
   - Arraylist → listImage;

Deskripsi :

   - Masukkan : namaProduk, hargaProduk, satuanProduk,
     deskripsiProduk, listImage
   - Proses :
     1. Inisialisasi nama method simpanproduk
     2. Melakukan cek kondisi dari method cekKolomIsian,
        apabila true maka akan melakukan proses berikut :
        a. Melakukan cek koneksi internet, apabila koneksi
           tersambung maka melakukan langkah berikut :
```

```
            i.  Mencoba untuk melakukan beberapa langkah
                sebagai berikut :
                1. Melakukan cek listImage, apabila nilai
                   sama dengan 0 maka melakukan langkah
                   berikut :
                        a. Menampilkan pesan peringatan bahwa
                           harus memilih minimal 1 foto
                2. Apabila nilai tidak sama dengan 0 maka
                   melakukan langkah berikut :
                        a. Melakukan set nilai pada class
                           model
                        b. Memanggil method buatProdukBaru
                           dengan nilai produkModel dan
                           listImage
           ii.  Jika terjadi kesalahan maka akan melakukan
                langkah berikut :
                 1. Menampilkan pesan error pada snackbar
        b. Jika kondisi intenet tidak tersambung maka
           melakukan langkah berikut :
            i. Menampilkan pesan pada snackbar bahwa tidak
               ada koneksi internet
    3. Apabila nilai cekKolomIsian false maka melakukan
       langkah sebagai berikut :
        a. Menampilkan pesan peringatan bahwa form isian masih
           kosong
```

Gambar 5.25 Algoritme Method Simpan Produk

5.1.5.3 Algoritme Buat Produk Baru

Algoritme `buat produk baru` merupakan algoritme yang berguna untuk melakukan proses simpan data produk ke dalam *database* produk reguler. Implementasi algoritma untuk `buat produk baru` dijelaskan pada Gambar 5.26

```
Algoritme buatProdukBaru
Deklarasi :
   - String → idProduk
   - HashMap → dataProduk, gambarProduk
Deskripsi :
   - Masukkan :
   - Proses :
      1. Inisialisasi nama method dengan nilai produkModel dan
         arraylist imageUri
      2. Mendapatkan key dari firebase database dan disimpan
         pada variabel idProduk
      3. Mendapatkan nilai dari beberapa atribut yang di
         peroleh dari class model dan disimpan pada konstanta
         tertentu dari varibael dataProduk
      4. Menyimpan nilai dari variabel dataProduk ke database
         firebase, jika proses sukses maka akan melakukan
         langkah sebagai berikut :
```

```
            a. Memanggil method uploadPhotoThreadListener
            b. Melakukan set nilai image url pada variabel
               gambarProduk
            c. Menyimpan nilai dari variabel gambarProduk ke
               database firebase, apabila proses sukses maka
               melakukan langkah sebagai berikut :
                i. Menampilkan pesan dalam bentuk Toast bila data
                   tersimpan
            d. Jika proses gagal maka melakukan langkah sebagai
               berikut :
                i. Menampilkan pesan dalam bentuk Toast bila
                   proses upload data foto gagal
      5.  Jika proses menyimpan dataProduk gagal maka melakukan
          langkah sebagai berikut :
                a. Menampilkan pesan dalam bentuk Toast bila
                   proses simpan data produk gagal
```

Gambar 5.26 Algoritme Method Buat Produk Baru

5.1.5.4 Algoritme Buat Produk Khusus

Algoritme `buat produk khusus` merupakan algoritme yang berguna untuk melakukan proses mendapatkan nilai variabel produk dari `edit text` yang kemudian di simpan dalam *database* produk khusus. Implementasi algoritma untuk `buat produk khusus` dijelaskan pada Gambar 5.27

```
Algoritme buatProdukKhusus
Deklarasi :

   -  HashMap → dataProduk
   -  String → namaProduk, satuanProduk, produkId
Deskripsi :

   -  Masukkan : inputNamaProduk, inputSatuanDigit
   -  Proses :
      1.  Inisialisasi nama method buatProdukKhusus
      2.  Melakukan cek kondisi dari method cekKolomIsian,
          apabila true maka melakukan proses sebagai berikut :
            a. Mendapatkan nilai dari inputNamaProduk dan disimpan
               ke dalama variabel namaProduk
            b. Mendapatkan nilai dari inputSatuanDigit dan
               disimpan ke dalama variabel satuanProduk
            c. Melakukan cek koneksi internet, apabila koneksi
               tersambung maka melakukan langkah berikut :
                i. Mencoba untuk melakukan beberapa langkah
                   sebagai berikut :
                    1. Mendapatkan push Id dan di simpan kedalam
                       variabel produkId
                    2. Menyimpan nilai variabel produkId,
                       namaProduk dan satuanProduk kedalam
                       variabel dataProduk
                    3. Menyimpan informasi nilai dari dataProduk
                       kedalam child ProdukKhusus, apabila proses
```

```
                    simpan produk sukses maka melakukan langkah
                    sebagai berikut :
                     a. Onsu
                     b. onfail
                ii. Jika terjadi kesalahan maka akan melakukan
                    langkah berikut :
                     1. Menampilkan pesan error pada snackbar
            d. Jika kondisi intenet tidak tersambung maka
               melakukan langkah berikut :
                i. Menampilkan pesan pada snackbar bahwa tidak ada
                   koneksi internet
    3. Apabila cek kondisi dari method cekKolomIsian false
       maka melakukan proses sebagai berikut :
        a. Menampilkan pesan peringatan bahwa form isian masih
           kosong
```

Gambar 5.27 Algoritme Method Buat Produk Khusus

5.1.5.5 Algoritme Pesan Produk Khusus

Algoritme `pesan produk khusus` merupakan algoritme yang berguna untuk melakukan proses mendapatkan nilai variabel pemesanan dari `edit text` yang kemudian di simpan dalam *database* pemesanan produk. Implementasi algoritma untuk `pesan produk khusus` dijelaskan pada Gambar 5.28

```
Algoritme pesanProdukKhusus

Deklarasi :

   - HashMap → pemesanan
   - String → idProduk, pesanKonsumen, tanggalKirim,
     waktuKirim, pemesananId
   - long → orderTime
Deskripsi :

   - Masukkan : notePenjual, inputTanggalKirim, inputJamKirim,
     inputJumlahProduk
   - Proses :
     1. Melakukan inisialisasi nama method pesanProdukKhusus
        dengan nilai produkId
     2. Mendapatkan nilai dari notePenjual dan disimpan ke
        dalama variabel pesanKonsumen
     3. Mendapatkan nilai dari inputTanggalKirim dan disimpan
        ke dalama variabel tanggalKirim
     4. Mendapatkan nilai dari inputJamKirim dan disimpan ke
        dalama variabel waktuKirim
     5. Melakukan cek koneksi internet, apabila koneksi
        tersambung maka melakukan langkah berikut :
         a. Mencoba untuk melakukan beberapa langkah sebagai
            berikut :
             i. Mendapatkan push Id dan di simpan kedalam
                variabel transaksiId
```

```
            ii. Menyimpan nilai idProduk, pesanKonsumen,
                tanggalKirim, waktuKirim, pemesanId, orderTime,
                dan jumlahProduk kedalam variabel pemesanan
            iii. Menyimpan informasi nilai dari pemesanan
                kedalam child Konsumen dan Penjual, apabila
                proses sukses maka melakukan langkah berikut :
                1. Menampilkan pesan dalam bentuk Toast bila
                   proses simpan data pemesanan sukses
            iv. Apabila proses simpan nilai pemesanan gagal,
                maka melakukan langkah berikut :
                1. Menampilkan pesan dalam bentuk Toast bila
                   proses simpan data pemesanan sukses
        b. Jika terjadi kesalahan maka akan melakukan langkah
           berikut :
            i. Menampilkan pesan error pada snackbar
    6. Jika kondisi intenet tidak tersambung maka melakukan
       langkah berikut :
        a. Menampilkan pesan pada snackbar bahwa tidak ada
           koneksi internet
```

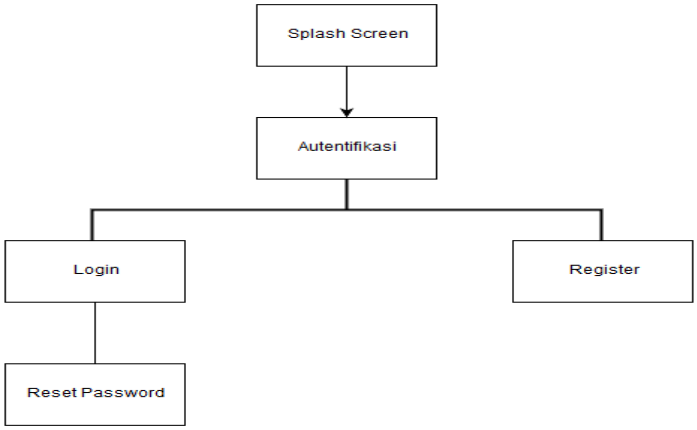
Gambar 5.28 Algoritme Method Pesan Produk Khusus

5.1.6 Perancangan Antarmuka

Perancangan antarmuka menjelaskan bagaimana rancangan dari antarmuka sistem aplikasi MLIJO. Perancangan antarmuka berisi tentang tampilan aplikasi yang akan digunakan oleh calon pengguna aplikasi. Pada perancangan antarmuka sistem, juga terbagi menjadi tiga yaitu perancangan antarmuka pada sisi sistem pengguna, pada sisi sistem konsumen dan pada sisi sistem penjual. *Screen flow* digunakan untuk menggambarkan keseluruhan antarmuka sistem yang sesuai dengan kebutuhan aktor yang terbagi menjadi pengguna, konsumen dan penjual.

5.1.6.1 Perancangan antarmuka pada sisi pengguna

Antarmuka pada sisi pengguna berupa halaman yang disediakan bagi pengguna aplikasi untuk dapat masuk kedalam sistem. Alur proses jalannya aplikasi di gambarkan oleh Screen flow 5.29 berikut :

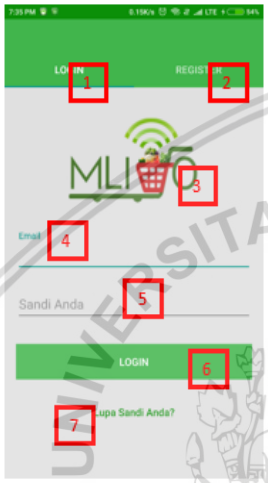


Gambar 5.29 Screen flow aplikasi pada sisi pengguna

Antarmuka untuk sisi pengguna terdiri dari antarmuka halaman *splashscreen*, halaman *tab login* dan halaman *tab* registrasi yang merupakan bagian dari activity autentifikasi dan halaman *reset password*. Pada saat aplikasi dijalankan, akan menampilkan halaman splashscreen terlebih dahulu kemudian berpindah ke halaman autentifikasi yang terdiri dari tab login dan register. Pada halaman login terdapat menu untuk berpindah menuju halaman reset password.

a. Halaman Login

Halaman login merupakan salah satu antarmuka pengguna yang berfungsi untuk masuk atau login kedalam sistem. Antarmuka login ditunjukan oleh gambar 5.30 berikut :

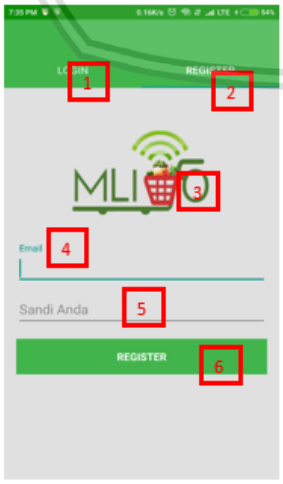


Keterangan :
1. Tab Login untuk berpindah ke halaman login.
2. Tab Register untuk berpindah ke halaman register
3. Logo aplikasi MLIJO
4. Input text untuk input email
5. Input text untuk input sandi
6. Tombol login, untuk menjalankan fungsi login sistem
7. Textview lupa sandi untuk pindah ke halaman lupa sandi

Gambar 5.30 Tampilan Antarmuka Halaman *Login*

b. Halaman *Register*

Halaman *register* merupakan salah satu antarmuka pengguna yang berfungsi untuk melakukan registrasi pengguna baru agar data pengguna terdaftar dalam sistem. Antarmuka *register* ditunjukan oleh gambar 5.31 berikut :

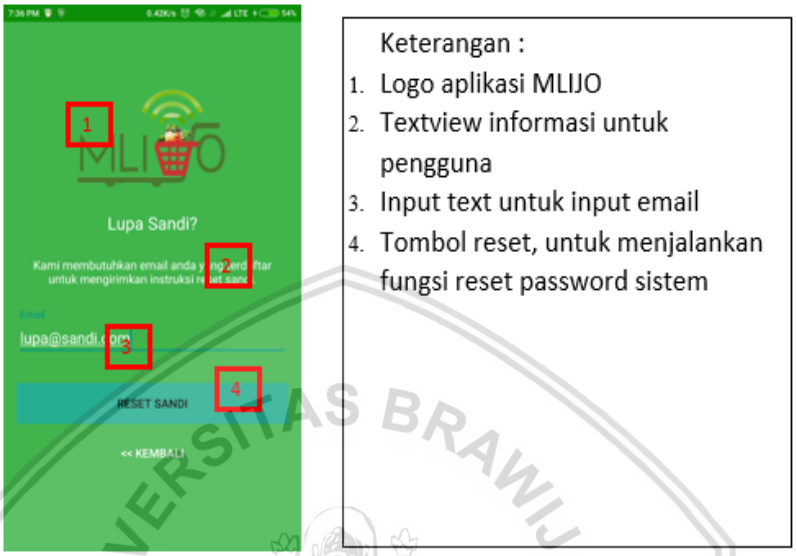


Keterangan :
1. Tab Login untuk berpindah ke halaman login.
2. Tab Register untuk berpindah ke halaman register
3. Logo aplikasi MLIJO
4. Input text untuk input email
5. Input text untuk input sandi
6. Tombol login, untuk menjalankan fungsi login sistem

Gambar 5.31 Tampilan Antarmuka Halaman *Register*

c. Halaman *Reset Password*

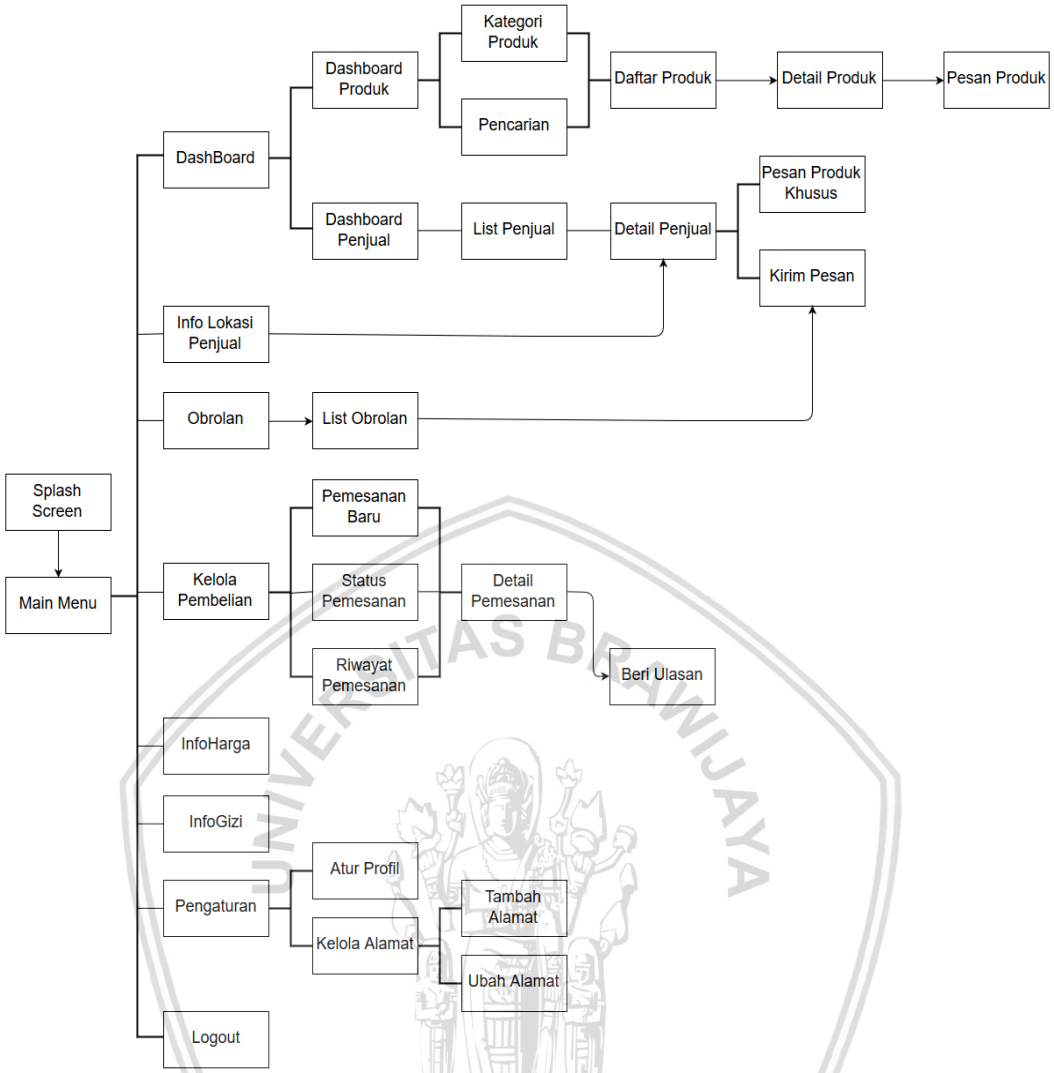
Halaman *reset password* merupakan salah satu antarmuka pengguna yang berfungsi untuk melakukan setel ulang *password* jika pengguna lupa terhadap *password* yang digunakan untuk login. Antarmuka *reset password* ditunjukan oleh gambar 5.32 berikut :



Gambar 5.32 Tampilan Antarmuka Halaman Reset Password

5.1.6.2 Perancangan antarmuka pada sisi konsumen

Antarmuka pada sisi konsumen berupa halaman yang disediakan bagi konsumen untuk dapat menggunakan semua fungsionalitas sistem. Antarmuka untuk sisi konsumen terdiri dari antarmuka halaman *splashscreen*, halaman *dashboard*, info lokasi penjual, obrolan, kelola pembelian, info harga, info gizi, pengaturan dan halaman lainnya. Screen flow aplikasi pada sisi konsumen ditunjukkan oleh Gambar 5.33 berikut :

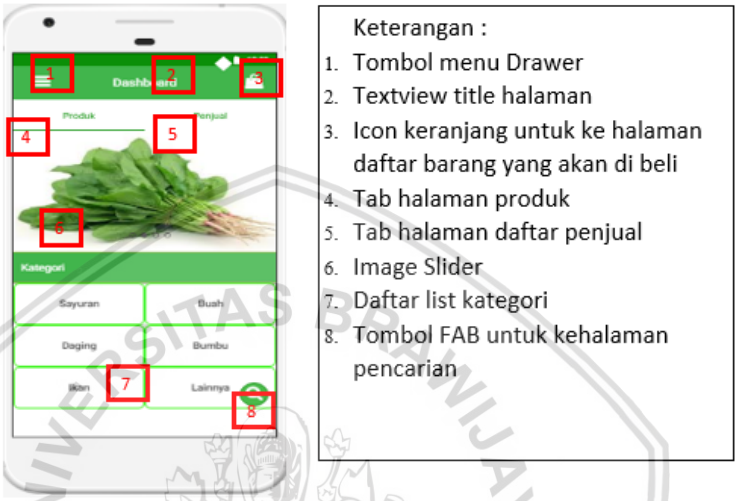


Gambar 5.33 Screen flow Aplikasi sisi Konsumen

Antarmuka untuk sisi konsumen terdiri dari antarmuka halaman *splashscreen*, halaman *dashboard*, info lokasi penjual, obrolan, kelola pembelian, info harga, info gizi, pengaturan dan halaman lainnya. Pada saat aplikasi dijalankan, akan menampilkan halaman splashscreen terlebih dahulu kemudian berpindah ke halaman main activity yang terdiri dari halaman yang telah disebutkan sebelumnya. Halaman main activity tersebut secara default merupakan halaman dashboard. Halaman dashboard terdiri dari tab produk dan penjual. Tab produk merupakan halaman yang berhubungan dengan segala sesuatu mengenai produk, mulai dari daftar produk dst. Sedangkan tab penjual merupakan terdiri dari daftar penjual dst. Menu selanjutnya adalah info lokasi penjual, yang mana terdapat tombol menuju halaman detail penjual di dalamnya. Selanjutnya halaman obrolan yang terdiri dari daftar obrolan. Kemudian halaman kelola pembelian yang terdiri dari tiga menu untuk menuju halaman daftar pembelian berdasarkan status yang ada. Kemudian terdapat halaman pengaturan, yang terdiri dari menu atur profil dan kelola alamat.

a. Halaman *Dashboard* Produk

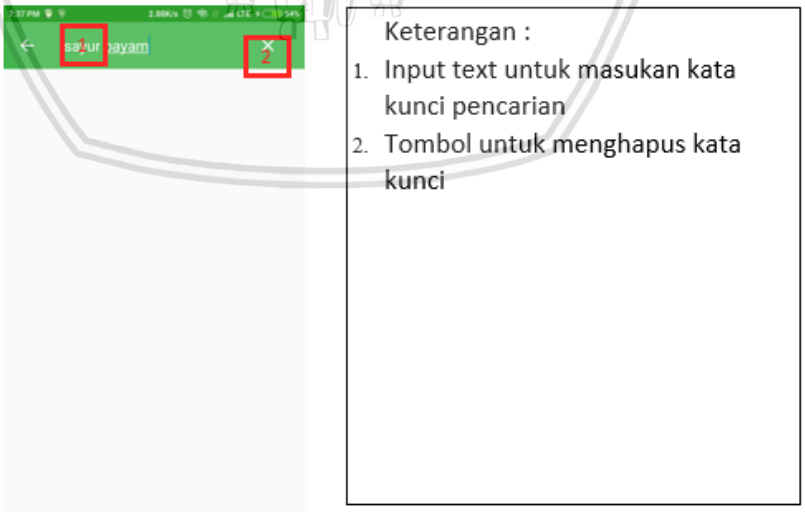
Halaman *dashboard* produk merupakan salah satu antarmuka konsumen yang merupakan halaman utaman konsumen. Halaman *dashboard* produk berfungsi untuk melakukan pencarian produk berdasarkan daftar kategori yang tersedia atau dengan kata kunci pada halaman pencarian yang dapat di akses oleh tombol FAB. Antarmuka *dashboard* produk ditunjukan oleh gambar 5.34 berikut :



Gambar 5.34 Tampilan Antarmuka Halaman *Dashboard* Produk

b. Halaman Input Pencarian

Halaman Input Pencarian merupakan salah satu antarmuka konsumen yang berfungsi untuk melakukan pencarian produk berdasarkan kata kunci. Antarmuka pencarian ditunjukan oleh gambar 5.35 berikut :



Gambar 5.35 Tampilan Antarmuka Halaman Pencarian

c. Halaman Daftar Produk

Halaman daftar produk merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan list daftar produk sesuai dengan kategori terpilih maupun pencarian kata kunci. Antarmuka daftar produk ditunjukan oleh gambar 5.36 berikut :

Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Icon keranjang belanja, untuk ke halaman daftar barang yang akan di beli
4. Tombol atur, untuk melakukan pengurutan tampilan
5. Tombol filter, untuk melakukan filter pencarian
6. Listview daftar produk

Gambar 5.36 Tampilan Halaman Antarmuka Daftar Produk

d. Halaman Detail Produk

Halaman Detail Produk merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan detail dari sebuah produk. Antarmuka detail produk ditunjukan oleh gambar 5.37 berikut :

Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Icon keranjang belanja, untuk ke halaman daftar barang yang akan di beli
4. Textview harga dan nama produk
5. Slide gambar produk
6. Textview deskripsi produk
7. Dropdown jumlah produk yang akan di pesan
8. Tombol untuk melakukan pemesanan produk

Gambar 5.37 Tampilan Antarmuka Halaman Detail Produk

e. Halaman Pesan Produk

Halaman Pesan Produk merupakan salah satu antarmuka konsumen yang berfungsi untuk melakukan pemesanan produk. Antarmuka pesan produk ditunjukan oleh gambar 5.38 berikut :

Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Icon keranjang belanja, untuk ke halaman daftar barang yang akan di beli
4. Imageview gambar produk
5. Textview nama produk
6. Textview jumlah produk
7. Inputtext untuk memberikan pesan kepada penjual
8. Textview alamat pengiriman
9. Tombol pilih alamat untuk memilih alamat tujuan pengiriman
10. Tombol tambah alamat untuk input alamat baru tujuan pengiriman
11. Textview keterangan total harga dan biaya pengiriman
12. Tombol pesan, untuk pesan produk

Gambar 5.38 Tampilan Antarmuka Halaman Pesan Produk

f. Halaman *Dashboard* Penjual

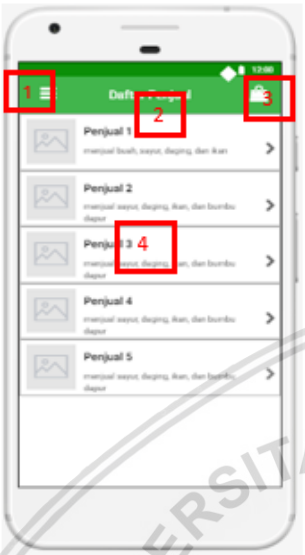
Halaman *dashboard* penjual merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan beberapa list penjual terbaru dan terdapat tombol untuk menuju ke halaman daftar penjual. Antarmuka *dashboard* penjual ditunjukan oleh gambar 5.39 berikut :

Keterangan :
1. Tombol menu Drawer
2. Textview judul halaman
3. Icon keranjang belanja, untuk ke halaman daftar barang yang akan di beli
4. Tablayout halaman produk untuk menuju halaman produk
5. Tablayout halaman penjual untuk menuju halaman penjual
6. Slide image informasi
7. Tombol daftar penjual untuk menuju list daftar penjual
8. Listview penjual terbaru

Gambar 5.39 Tampilan Antarmuka Halaman *Dashboard* Penjual

g. Halaman Daftar Penjual

Halaman daftar penjual merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan daftar list penjual. Antarmuka daftar penjual ditunjukan oleh gambar 5.40 berikut :



Keterangan :
1. Tombol menu Drawer
2. Textview judul halaman
3. Icon keranjang belanja, untuk ke halaman daftar barang yang akan di beli
4. Listview daftar nama penjual

Gambar 5.40 Tampilan Antarmuka Halaman Daftar Penjual

h. Halaman Detail Penjual

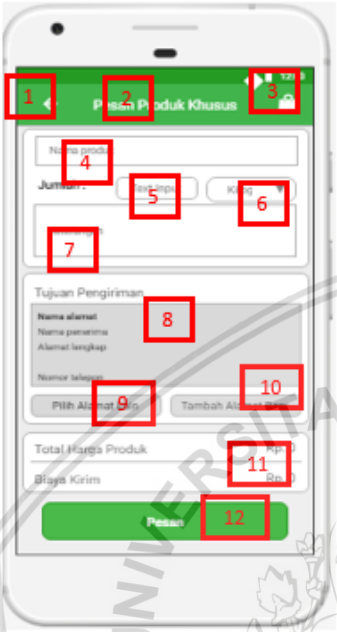
Halaman detail penjual merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan detail dari profil penjual. Selain itu pada halaman tersebut terdapat tombol untuk menuju halaman pesan produk khusus dan halaman kirim obrolan. Antarmuka detail profil penjual ditunjukan oleh gambar 5.41 berikut :

Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Imageview avatar penjual
4. Textview nama penjual
5. Gridview informasi kategori produk dari penjual
6. Ratingbar indicator penilaian penjual
7. Textview informasi kontak dan alamat penjual
8. Tombol pesan produk untuk melakukan pesan produk khusus
9. Tombol kirim pesan untuk melakukan kirim pesan(chat)

Gambar 5.41 Tampilan Antarmuka Halaman Detail Penjual

i. Halaman Pesan Produk Khusus

Halaman pesan produk khusus merupakan salah satu antarmuka konsumen yang berfungsi untuk melakukan pemesanan produk secara khusus terhadap penjual tertentu. Antarmuka pesan produk khusus ditunjukan oleh gambar 5.42 berikut :

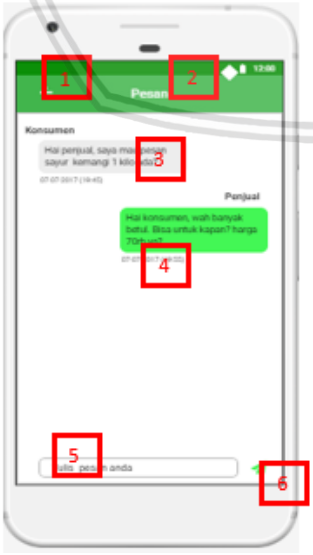


Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Icon keranjang belanja, untuk ke halaman daftar barang yang akan di beli
4. Input text untuk input nama produk
5. Input text untuk input nominal satuan produk
6. Dropdown satuan untuk memilih satuan nominal produk
7. Input text untuk input keterangan pemesanan kepada penjual
8. Textview informasi alamat tujuan pengiriman
9. Tombol pilih alamat untuk memilih alamat tujuan pengiriman
10. Tombol tambah alamat untuk input alamat baru tujuan pengiriman
11. Textview keterangan total harga dan biaya pengiriman
12. Tombol pesan untuk melakukan pemesanan produk

Gambar 5.42 Tampilan Antarmuka Halaman Pesan Produk Khusus

j. Halaman Obrolan

Halaman obrolan merupakan salah satu antarmuka konsumen yang berfungsi untuk melakukan kirim obrolan dengan penjual tertentu. Antarmuka obrolan ditunjukan oleh gambar 5.43 berikut :



Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Textview chat dari pengirim
4. Textview chat dari penerima
5. Input text untuk menginput pesan
6. Icon untuk mengirim pesan yang telah di ketik

Gambar 5.43 Tampilan Antarmuka Halaman Obrolan

k. Halaman Peta Penjual

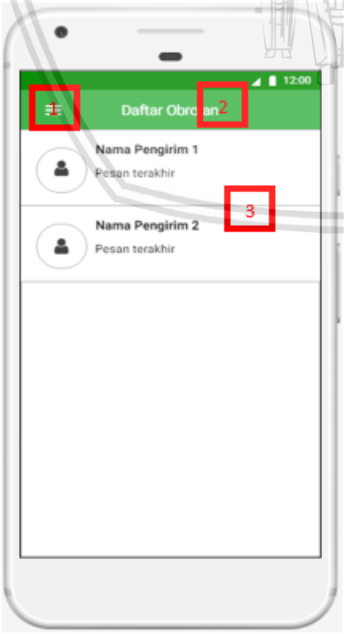
Halaman peta penjual merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan peta lokasi penjual aktif. Antarmuka peta penjual ditunjukan oleh gambar 5.44 berikut :

Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Peta tampilan lokasi penjual aktif

Gambar 5.44 Tampilan Antarmuka Halaman Peta Penjual

l. Halaman Daftar Obrolan

Halaman daftar obrolan merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan list obrolan. Antarmuka daftar obrolan ditunjukan oleh gambar 5.45 berikut :



Keterangan :
1. Tombol menu Drawer
2. Textview judul halaman
3. Listview daftar obrolan

Gambar 5.45 Tampilan Antarmuka Halaman Daftar Obrolan

m. Halaman Daftar Kelola Pembelian

Halaman daftar kelola pembelian merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan daftar menu pesanan baru, status pesanan dan riwayat pesanan. Antarmuka daftar kelola pembelian ditunjukan oleh gambar 5.46 berikut :

Keterangan :
1. Tombol menu Drawer
2. Textview judul halaman
3. Icon keranjang belanja, untuk ke halaman daftar barang yang akan di beli
4. Layout informasi pembelian terbaru dan juga untuk menuju halaman daftar pembelian baru
5. Layout informasi status pemesanan dan juga untuk menuju halaman daftar pemesanan yang telah di proses
6. Layout informasi riwayat tranaksi dan juga untuk menuju halaman daftar riwayat transaksi

Gambar 5.46 Tampilan Antarmuka Halaman Daftar Kelola Pembelian

n. Halaman Daftar Transaksi

Halaman daftar transaksi merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan list transaksi pembelian. Antarmuka daftar pembelian ditunjukan oleh gambar 5.47 berikut :

Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Listview daftar pembelian

Gambar 5.47 Tampilan Antarmuka Halaman Daftar Pembelian

o. Halaman Detail Transaksi

Halaman detail transaksi merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilan detail dari transaksi. Antarmuka detail transaksi ditunjukan oleh gambar 5.48 berikut :

Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Imageview gambar produk
4. Textview nama produk
5. Textview jumlah produk dan keterangan dari konsumen untuk penjual
6. Textview keterangan total harga dan biaya pengiriman
7. Textview alamat pengiriman
8. Textview informasi status terbaru dan nama penerima produk
9. Tombol untuk melakukan konfirmasi penerimaan produk

Gambar 5.48 Tampilan Antarmuka Halaman Detail Transaksi

p. Halaman Beri Ulasan

Halaman beri ulasan merupakan salah satu antarmuka konsumen yang berfungsi untuk memberikan ulasan kepada penjual setelah transaksi selesai. Antarmuka beri ulasan ditunjukan oleh gambar 5.49 berikut :

Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Imageview gambar produk
4. Textview nama produk
5. Ratingbar untuk input nilai kualitas produk
6. Textview nilai kualitas produk
7. Ratingbar untuk input nilai kualitas pelayanan
8. Textview nilai kualitas pelayanan
9. Inputtext untuk melakukan ulasan terhdap penjual
10. Tombol untuk mengirim ulasan

Gambar 5.49 Tampilan Antarmuka Halaman Beri Ulasan

q. Halaman Info Harga

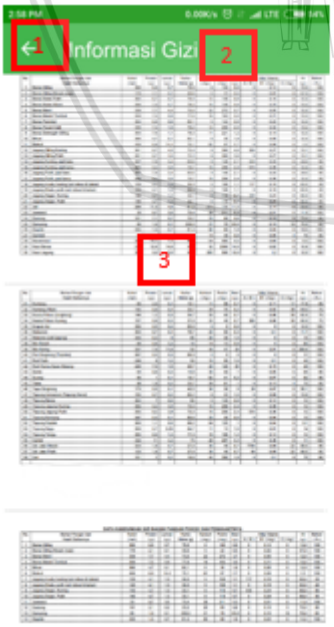
Halaman Info harga merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan tabel informasi harga terbaru setiap harinya. Antarmuka info harga ditunjukan oleh gambar 5.50 berikut :

Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Tabel informasi harga terkini

Gambar 5.50 Tampilan Antarmuka Halaman Info Harga

r. Halaman Info Gizi

Halaman Info gizi merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan tabel daftar informasi nilai gizi suatu makanan. Antarmuka info gizi ditunjukan oleh gambar 5.51 berikut :

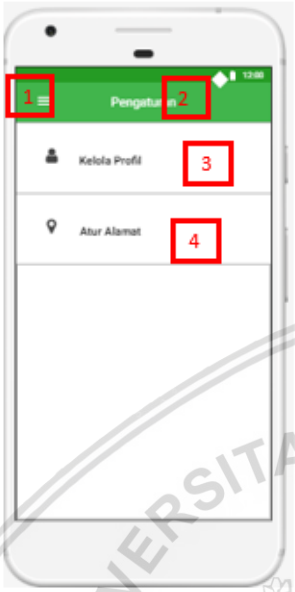


Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Tabel informasi nilai gizi

Gambar 5.51 Tampilan Antarmuka Halaman Informasi Gizi

s. Halaman Pengaturan

Halaman pengaturan merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan menu kelola profil. Antarmuka pengaturan ditunjukan oleh gambar 5.52 berikut :



Keterangan :
1. Tombol menu Drawer
2. Textview title halaman
3. Layout kelola profil untuk menuju halaman kelola profil
4. Layout atur alamat untuk menuju halaman pengaturan alamat

Gambar 5.52 Tampilan Antarmuka Halaman Pengaturan

t. Halaman Pengaturan Profil

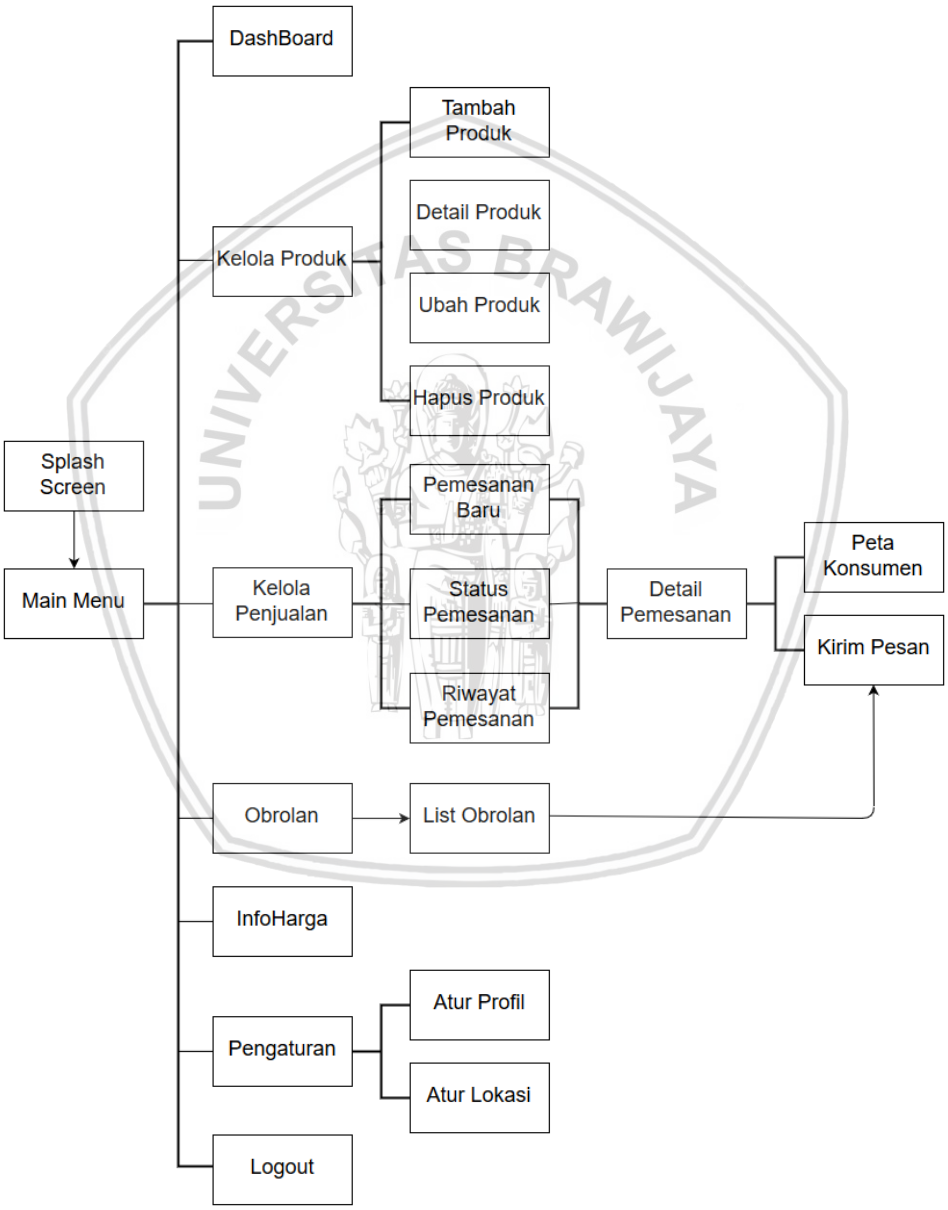
Halaman pengaturan profil merupakan salah satu antarmuka konsumen yang berfungsi untuk melakukan upload foto profil dan edit data profil. Antarmuka pengaturan profil ditunjukan oleh gambar 5.53 berikut :

Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Imageview avatar penjual dan juga berfungsi sebagai tombol untuk upload avatar
4. Inputtext untuk input nama
5. Datepicker untuk input tanggal lahir
6. Radio button untuk memilih jenis kelamin
7. Tombol simpan untuk menyimpan data profil

Gambar 5.53 Tampilan Antarmuka Halaman Pengaturan Profil

5.1.6.3 Perancangan antarmuka pada sisi penjual

Antarmuka pada sisi penjual berupa halaman yang disediakan bagi penjual untuk dapat menggunakan semua fungsionalitas sistem pada sisi penjual. Antarmuka untuk sisi penjual terdiri dari antarmuka halaman splashscreen, halaman *dashboard*, kelola produk, kelola penjualan, obrolan, info harga, pengaturan dan halaman lainnya. Beberapa fungsi antarmuka pada sisi penjual tersebut sama seperti pada sisi konsumen, seperti kelola transaksi, obrolan dan info harga, sehingga penulis tidak menuliskan kembali pada sub bab berikut. Screen flow aplikasi pada sisi penjual ditunjukkan oleh Gambar 5.54 berikut :



Gambar 5.54 Screen flow Aplikasi sisi Penjual

Antarmuka untuk sisi penjual terdiri dari antarmuka halaman *splashscreen*, halaman *dashboard*, kelola produk, kelola penjualan, obrolan, info harga, pengaturan dan halaman lainnya. Pada saat aplikasi dijalankan, akan menampilkan

halaman splashscreen terlebih dahulu kemudian berpindah ke halaman main activity yang terdiri dari beberapa halaman yang telah disebutkan sebelumnya. Halaman main activity tersebut secara default merupakan halaman dashboard. Selanjutnya menu kelola produk, dimana halaman tersebut terdiri dari beberapa halaman seperti tambah produk dan lainnya. Kemudian halaman kelola penjualan yang terdiri dari tiga menu untuk menuju halaman daftar penjualan berdasarkan status yang ada. Selanjutnya halaman obrolan yang terdiri dari daftar obrolan.Kemudian terdapat halaman pengaturan, yang terdiri dari menu atur profil dan kelola alamat.

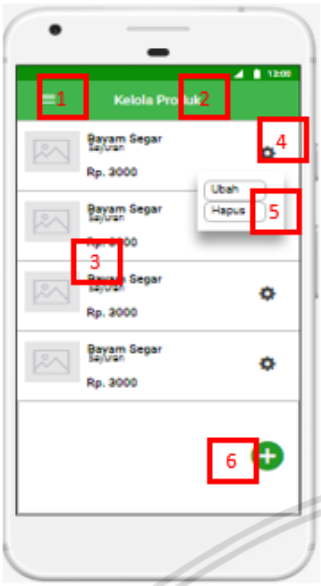
a. Halaman *Dashboard*

Halaman *dashboard* merupakan salah satu antarmuka penjual yang berfungsi untuk menampilkan status berjualan penjual dan tombol switch untuk merubah status tersebut. Antarmuka *dashboard* ditunjukan oleh gambar 5.55 berikut :

Gambar 5.55 Tampilan Antarmuka Halaman *Dashboard*

b. Halaman Kelola Produk

Halaman kelola produk merupakan antarmuka penjual yang berfungsi untuk menampilkan daftar produk beserta menu kelola dan tambah produk. Antarmuka kelola produk ditunjukan oleh gambar 5.56 berikut:

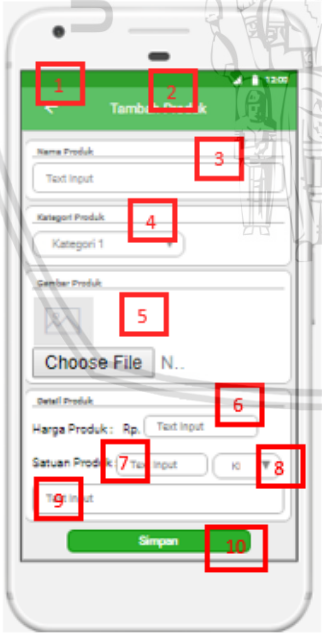


Keterangan :
1. Tombol menu Drawer
2. Textview judul halaman
3. Listview daftar produk
4. Tombol menu untuk menampilkan dialog menu
5. Dialog box berisi menu ubah dan hapus produk
6. Tombol FAB untuk melakukan tambah produk baru

Gambar 5.56 Tampilan Antarmuka Halaman Kelola Produk

c. Halaman Tambah Produk

Halaman tambah produk merupakan salah satu antarmuka penjual yang berfungsi untuk membuat data produk baru. Antarmuka tambah produk ditunjukan oleh gambar 5.57 berikut :

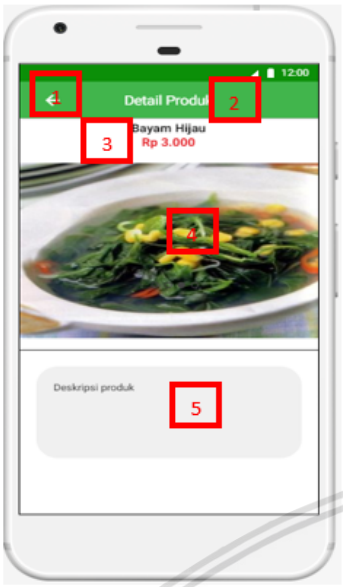


Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Textinput untuk input nama produk
4. Dropdown menu untuk memilih kategori produk
5. Tombol upload, untuk melakukan upload foto
6. Textinput untuk input harga produk
7. Textinput untuk input nominal satuan produk
8. Dropdown satuan untuk memilih satuan produk
9. Textinput untuk input deskripsi produk
10. Tombol simpan untuk menyimpan data produk

Gambar 5.57 Tampilan Antarmuka Halaman Tambah Produk

d. Halaman Detail Produk

Halaman detail produk merupakan salah satu antarmuka penjual yang berfungsi untuk menampilkan detail data dari sebuah produk. Antarmuka detail produk ditunjukan oleh gambar 5.58 berikut :

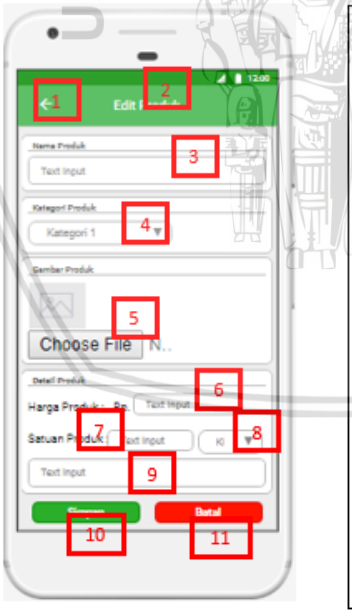


Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Textview nama produk dan harga produk
4. Image slideview produk
5. Textview deskripsi produk

Gambar 5.58 Tampilan Antarmuka Halaman Detail Produk

e. Halaman Ubah Produk

Halaman ubah produk merupakan salah satu antarmuka penjual yang berfungsi untuk melakukan perubahan data pada produk tertentu. Antarmuka ubah produk ditunjukan oleh gambar 5.59 berikut :



Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Textinput untuk input nama produk
4. Dropdown menu untuk memilih kategori produk
5. Tombol upload, untuk melakukan upload foto
6. Textinput untuk input harga produk
7. Textinput untuk input nominal satuan produk
8. Dropdown satuan untuk memilih satuan produk
9. Textinput untuk input deskripsi produk
10. Tombol untuk menyimpan hasil edit data produk
11. Tombol batal untuk batal edit data

Gambar 5.59 Tampilan Antarmuka Halaman Ubah Produk

f. Halaman Detail Transaksi

Halaman detail transaksi merupakan salah satu antarmuka penjual yang berfungsi untuk menampilan detail dari transaksi. Antarmuka detail transaksi ditunjukan oleh gambar 5.60 berikut :

Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Imageview gambar produk
4. Textview nama produk
5. Textview jumlah produk dan keterangan dari konsumen untuk penjual
6. Textview keterangan total harga dan textinput biaya pengiriman
7. Textview alamat pengiriman
8. Tombol untuk melakukan kirim pesan ke konsumen
9. Tombol untuk melihat peta menuju lokasi konsumen
10. Tombol untuk menerima pesanan konsumen
11. Tombol untuk menolak pesanan konsumen

Gambar 5.60 Tampilan Antarmuka Halaman Detail Transaksi

g. Halaman Peta Konsumen

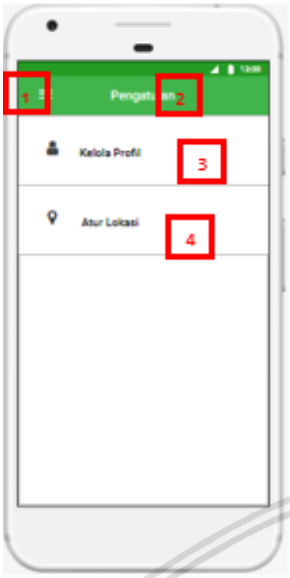
Halaman peta konsumen merupakan salah satu antarmuka penjual yang berfungsi untuk manampilkan peta lokasi konsumen yang melakukan pemesanan. Antarmuka peta konsumen ditunjukan oleh gambar 5.61 berikut:

Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Peta tampilan lokasi alamat konsumen
4. Tombol GoogleMaps berfungsi menuju ke aplikasi google maps, untuk menunjukan rute dari lokasi penjual ke alamat konsumen

Gambar 5.61 Tampilan Antarmuka Halaman Peta Konsumen

h. Halaman Pengaturan

Halaman pengaturan merupakan salah satu antarmuka penjual yang berfungsi untuk menampilkan menu kelola profil dan menu atur lokasi. Antarmuka pengaturan ditunjukan oleh gambar 5.62 berikut :

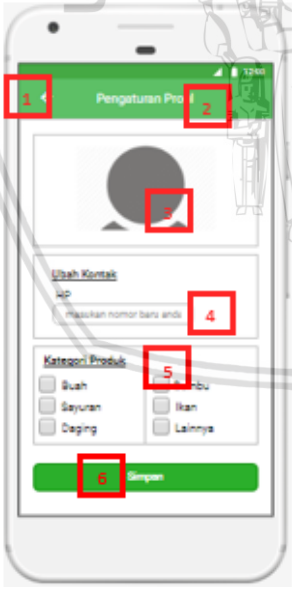


Keterangan :
1. Tombol menu Drawer
2. Textview title halaman
3. Layout kelola profil untuk menuju halaman kelola profil
4. Layout atur lokasi untuk menuju halaman pengaturan lokasi penjual

Gambar 5.62 Tampilan Antarmuka Halaman Pengaturan

i. Halaman Pengaturan Profil

Halaman pengaturan profil merupakan salah satu antarmuka penjual yang berfungsi untuk melakukan upload foto profil, edit nomor telepon, dan data kategori produk yang dijual. Antarmuka pengaturan profil ditunjukan oleh gambar 5.63 berikut :



Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Imageview avatar penjual dan juga berfungsi sebagai tombol untuk upload avatar
4. Inputtext untuk input nomor hp penjual
5. Checkbox kategori produk
6. Tombol simpan untuk menyimpan data profil

Gambar 5.63 Tampilan Antarmuka Halaman Pengaturan Profil

j. Halaman Pengaturan Lokasi

Halaman pengaturan lokasi merupakan salah satu antarmuka penjual yang berfungsi untuk mengatur info area berjualan penjual, waktu berjualan dan juga fitur berbagi lokasi. Antarmuka pengaturan lokasi ditunjukan oleh gambar 5.64 berikut :

Keterangan :
1. Tombol kembali ke halaman sebelumnya
2. Textview judul halaman
3. Textinput untuk melakukan input area berjualan
4. Textinput untuk melakukan input jam berjualan
5. Textview status lokasi penjual
6. Tombol switch untuk merubah status lokasi penjual
7. Tombol simpan untuk menyimpan perubahan data

Gambar 5.64 Tampilan Antarmuka Halaman Lokasi

5.2 Implementasi

Bab ini membahas mengenai tahapan implementasi perangkat lunak Mlijo berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan proses perancangan perangkat lunak. Pembahasan terdiri atas penjelasan tentang spesifikasi sistem, batasan – batasan dalam implementasi, implementasi basis data, implementasi tiap *class* pada file program, implementasi kode program, dan implementasi antarmuka.

5.2.1 Spesifikasi Sistem

Perangkat lunak Mlijo dikembangkan dalam lingkungan implementasi yang terdiri dari perangkat keras dan perangkat lunak.

5.2.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan dalam proses pengembangan aplikasi Mlijo dijelaskan pada tabel 5.12 berikut :

Tabel 5.12 Spesifikasi Perangkat Keras komputer

Notebook Asus A451LN	
Processor	Intel (R) Core(TM) i5-4200U 1.60GHz
Memory(RAM)	8 GB
Harddisk	Seagate ST1000LM024 HN-M101MBB 5400 SATA 6Gb/s 1TB
Motherboard	ASUS S451LN Intel Lynx Point-LP (Premium)
Graphic Card	NVIDIA GeForce GT 840M
Monitor	14” N140BGE-E43 CMN

5.2.1.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam proses pengembangan aplikasi Mlijo dijelaskan pada tabel 5.13 berikut :

Tabel 5.13 Spesfikasi Perangkat Lunak Komputer

Notebook Asus A451LN	
Operating System	Microsoft Windows 10 Pro 64bit
Programming Language	Java
Programming Tool	Android Studio 3.0.1
Android SDK Tools	26.1.1
Gradle version	4.1
Database Management System	Firebase *database* SDK 11.4.2

5.2.1.3 Spesifikasi Perangkat Bergerak Sisi Konsumen

Spesifikasi perangkat bergerak yang digunakan dalam proses pengembangan apliksi Mlijo dijelaskan pada tabel 5.14 berikut :

Tabel 5.14 Spesifikasi Perangkat Bergerak sisi Konsumen

Smartphone Android Xiaomi Redmi Note 3 Pro	
Operating System	Android marshmallow 6.0.1 API 23
CPU	Hexa-core (4x1.4 GHz Cortex-A53 & 2x1.8 GHz Cortex-A72)
GPU	Adreno 510
Memory(RAM)	2 GB
Memory(Storage)	16 GB
Chipset	Qualcomm MSM8956 Snapdragon 650
Display	5.5 inches, 1080 x 1920 pixels

5.2.1.4 Spesifikasi Perangkat Bergerak Sisi Penjual

Spesifikasi perangkat bergerak yang digunakan dalam proses pengembangan apliksi Mlijo dijelaskan pada tabel 5.15 berikut :

Tabel 5.15 Spesifikasi Perangkat Bergerak sisi Penjual

Emulator Android Nexus 5	
Operating System	Android kitkat 4.4 API 19
CPU	Multi-core 2

GPU	-
Memory(RAM)	1.5 GB
Memory(Storage)	1 GB
Chipset	-
Display	5 inches, 1080 x 1920 pixels

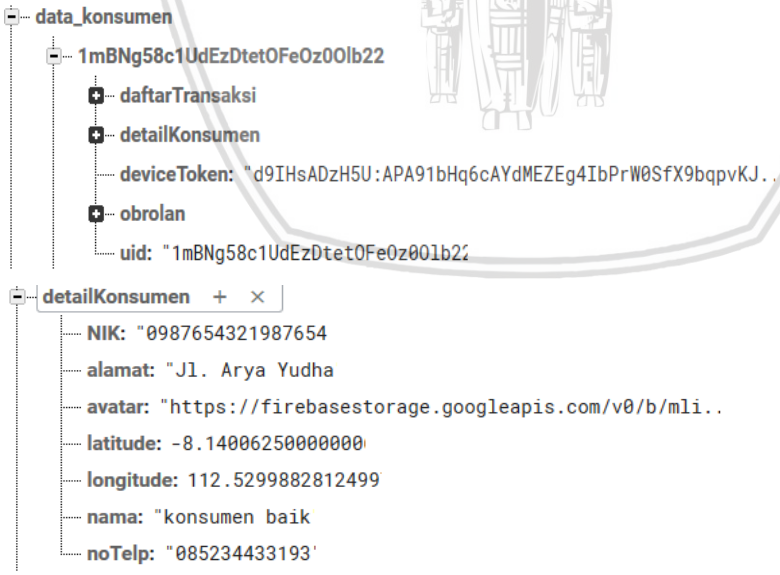
5.2.2 Batasan Implementasi

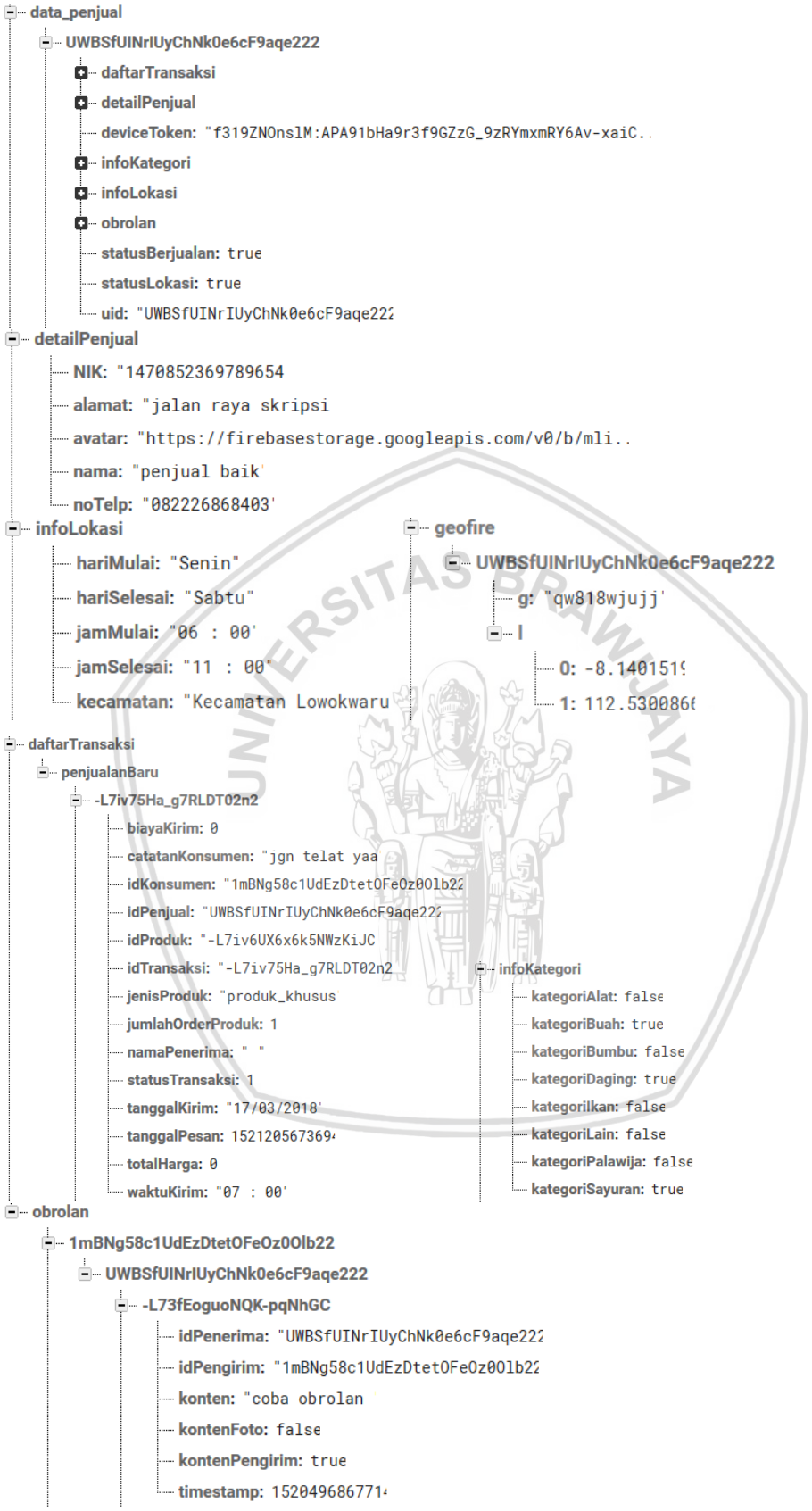
Beberapa batasan dalam mengimplementasikan perangkat lunak Mlijo adalah sebagai berikut :

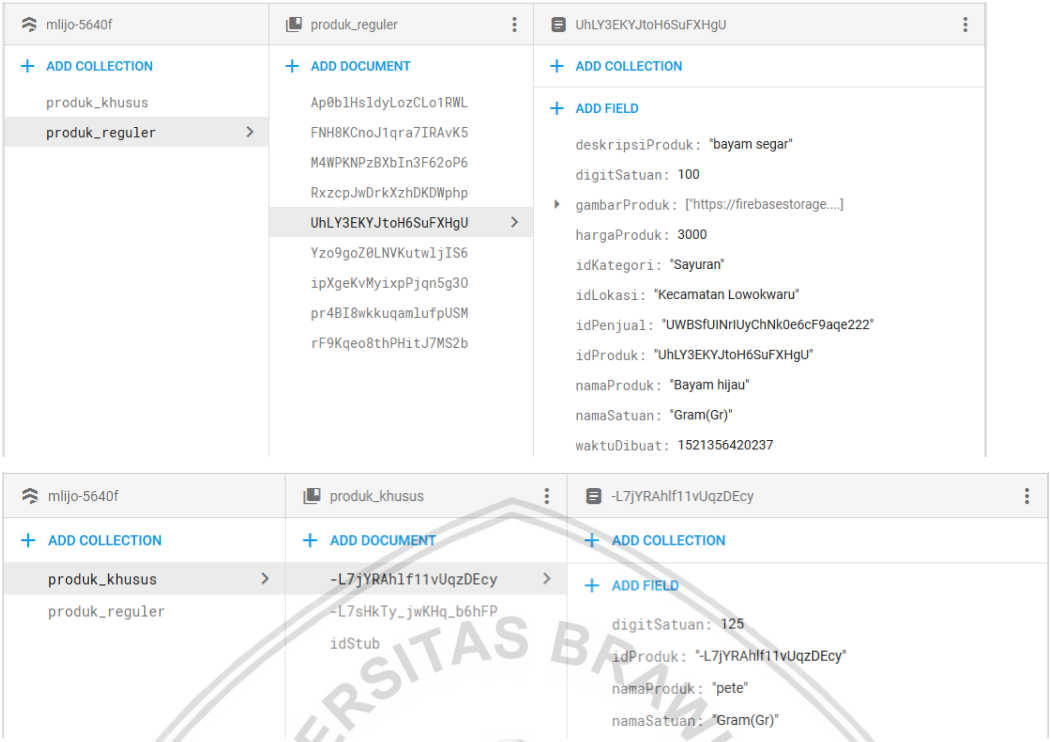
1. Perangkat lunak Mlijo dirancang dan diimplementasikan dengan menggunakan Android Studio.
2. Perangkat lunak Mlijo dibangun dengan bahasa pemograman Java.
3. Perangkat lunak Mlijo dirancang untuk perangkat android dengan API 19 keatas atau sistem operasi android Kitkat dan yang lebih baru.
4. Perangkat lunak Mlijo menggunakan *Firebase* sebagai manajemen basis data.

5.2.3 Implementasi *Database*

Implementasi penyimpanan data dilakukan dengan menggunakan *database firebase*. Hasil implementasi penyimpanan data ini berupa *script json*. Hasil implementasi json pada *database* ini ditunjukan dalam tabel berikut :




data_penjual
UWBSfUINrIUyChNk0e6cF9aqe222
daftarTransaksi
detailPenjual
deviceToken: "f319ZNOnslM:APA91bHa9r3f9GZzG_9zRYmxmRY6Av-xaiC..
infoKategori
infoLokasi
obrolan
statusBerjualan: true
statusLokasi: true
uid: "UWBSfUINrIUyChNk0e6cF9aqe222
detailPenjual
NIK: "1470852369789654
alamat: "jalan raya skripsi
avatar: "https://firebasestorage.googleapis.com/v0/b/mli..
nama: "penjual baik
noTelp: "082226868403"
infoLokasi
hariMulai: "Senin"
hariSelesai: "Sabtu"
jamMulai: "06 : 00"
jamSelesai: "11 : 00"
kecamatan: "Kecamatan Lowokwaru
geofire
UWBSfUINrIUyChNk0e6cF9aqe222
g: "qw818wjujj"
l
0: -8.140151
1: 112.530086
daftarTransaksi
penjualanBaru
-L7iv75Ha_g7RLDT02n2
biayaKirim: 0
catatanKonsumen: "jgn telat yaa
idKonsumen: "1mBNg58c1UdEzDtetOFeOz00lb22
idPenjual: "UWBSfUINrIUyChNk0e6cF9aqe222
idProduk: "-L7iv6UX6x6k5NWzKiJC
idTransaksi: "-L7iv75Ha_g7RLDT02n2
jenisProduk: "produk_khusus
jumlahOrderProduk: 1
namaPenerima: " "
statusTransaksi: 1
tanggalKirim: "17/03/2018"
tanggalPesan: 152120567369
totalHarga: 0
waktuKirim: "07 : 00"
infoKategori
kategoriAlat: false
kategoriBuah: true
kategoriBumbu: false
kategoriDaging: true
kategoriIkan: false
kategoriLain: false
kategoriPalawija: false
kategoriSayuran: true
obrolan
1mBNg58c1UdEzDtetOFeOz00lb22
UWBSfUINrIUyChNk0e6cF9aqe222
-L73fEoguoNQK-pqNhGC
idPenerima: "UWBSfUINrIUyChNk0e6cF9aqe222
idPengirim: "1mBNg58c1UdEzDtetOFeOz00lb22
konten: "coba obrolan
kontenFoto: false
kontenPengirim: true
timestamp: 152049686771



Gambar 5.65 Implementasi Database MLIJO pada Firebase Realtime Database

5.2.4 Implementasi *Class*

Setiap *class* yang telah dirancang pada proses perancangan direalisasikan pada sebuah file program dengan ekstensi *.java. Pasangan antara *class* dengan file program yang digunakan untuk implementasinya di tunjukan oleh tabel 5.16 untuk sistem konsumen dan tabel 5.17 untuk sistem penjual:

Tabel 5.16 implementasi *class* pada kode program *.java sistem konsumen

No.	Package	Nama *class* atau interface	Nama File program
1	Produk	DaftarProdukActivity	DaftarProdukActivity.java
2	Produk	DaftarProdukAdapter	DaftarProdukAdapter.java
3	Produk	DetailProdukActivity	DetailProdukActivity.java
4	Produk	KategoriModel	KategoriModel.java
5	Produk	ProdukModel	ProdukModel.java
6	Produk	PencarianActivity	PencarianActivity.java
7	Pesan Produk	PesanProdukActivity	PesanProdukActivity.java
8	Penjual	DaftarPenjualAdapter	DaftarPenjualAdapter.java
9	Penjual	DetailPenjualActivity	DetailPenjualActivity.java
10	Penjual	PenjualModel	PenjualModel.java

11	Penjual	PesanProdukKhususActivity	PesanProdukKhususActivity.java
12	Penjual	DaftarProdukPenjualActvity	DaftarProdukPenjualActivity.java

Tabel 5.17 Implementasi *class* pada kode program *.java sistem penjual

No.	Package	Nama *class* atau interface	Nama File program
1	KelolaProduk	KelolaProdukFragment	KelolaProdukFragment.java
2	KelolaProduk	KelolaProdukAdapter	KelolaProdukAdapter.java
3	KelolaProduk	ProdukModel	ProdukModel.java
4	KelolaProduk	TambahProdukActivity	TambahProdukActivity.java
5	KelolaProduk	UbahProdukActivity	UbahProdukActivity.java
6	KelolaProduk	postRefModel	PostRefModel.java
7	KelolaProduk	DetailProdukActivity	DetailProdukActivity.java
8	Kelola Penjualan	KelolaPenjualanFragment	KelolaPenjualanFragment.java
9	Kelola Penjualan	DaftarTransaksiAdapter	DaftarTransaksiAdapter.java
10	Kelola Penjualan	DaftarTransaksiActivity	DaftarTransaksiActivity.java
11	Kelola Penjualan	DetailPesananActivity	DetailPesananActivity.java
12	Kelola Penjualan	TransaksiModel	TransaksiModel.java
13	Obrolan	DaftarObrolanFragment	DaftarObrolanFragment.java
	Obrolan	DaftarObrolanAdapter	DaftarObrolanAdapter.java
14	Obrolan	ObrolanActivity	ObrolanActivity.java
15	Obrolan	ObrolanAdapter	ObrolanAdapter.java
16	Obrolan	ObrolanModel	ObrolanModel.java
17	Obrolan	ObrolanTerakhirModel	ObrolanTerakhirModel.java

5.2.5 Implementasi Kode Program

Implementasi kode program menjelaskan bagaimana hasil implementasi kode program dari aplikasi MLIJO yang telah dikembangkan berdasarkan fungsi sistem. Implementasi kode program perangkat lunak Mlijo terdiri dari 2 bagian, yaitu sisi konsumen dan sisi penjual yang ditulis dalam bahasa java.

5.2.5.1 Implementasi *Method* cekKolomIsian()

Method `cekKolomIsian` merupakan method yang berguna untuk melakukan pemeriksaan apakah kolom edit text yang tersedia telah diisi atau masih kosong. Method ini bernilai kembalian dengan tipe boolean. *Method* `cekKolomIsian` terdapat pada *method* `simpanProduk` sisi penjual maupun method `buatProdukKhusus` pada sisi konsumen.

```
1   private boolean cekKolomIsian() {
2       boolean hasil;
3       if (TextUtils.isEmpty(inputNamaProduk.getText()) ||
4   TextUtils.isEmpty(inputHargaProduk.getText()) ||
5   TextUtils.isEmpty(inputNominalSatuan.getText()) ||
6               TextUtils.isEmpty(inputDeskripsiProduk.getText())) {
7           hasil = false;
8       } else {
9           hasil = true;
10      }
11      return hasil;
12  }
```

Gambar 5.66 Implementasi Kode Program *Method* cekKolomIsian

Tabel 5.18 Penjelasan *Method* cekKolomIsian

Nomor Kode Program	Penjelasan
1	Inisialisasi nama *method* cekKolomIsian
2	Deklarasi variabel hasil dengan tipe data boolean
3 – 7	Kondisi if dengan set nilai variabel hasil = false
8 – 10	Kondisi else dengan set nilai variabel hasil = true
11	Mendapatkan nilai kembalian dari variable hasil
12	Kurung penutup dari *method* cekKolomIsian

5.2.5.2 Implementasi *Method* simpanProduk()

Method `simpanProduk` merupakan method yang berguna untuk melakukan perintah saat tombol buat produk baru pada sisi penjual di tekan. *Method* `simpanProduk` ini juga berfungsi memanggil method `buatProdukBaru` pada *class* presenter.

```
1   private void simpanProduk(){
2       if (cekKolomIsian() == true) {
3           namaProduk = inputNamaProduk.getText().toString();
4           hargaProduk =
5   Double.valueOf(inputHargaProduk.getText().toString());
6           satuanProduk =
7   Integer.parseInt(inputNominalSatuan.getText().toString());
8           deskripsiProduk =
9   inputDeskripsiProduk.getText().toString();
10          long waktuDibuat = new Date().getTime();
11          if
```

```
12  (InternetConnection.getInstance().isOnline(BuatProdukActivity.th
13  is)) {
14              try {
15                  if (listImage.size() == 0) {
16                      ShowAlertDialog.showAlert("Anda harus
17  memilih gambar produk minimal (1)!", this);
18                  } else {
19                      showProgessDialog();
20                      ProdukModel produkModel = new
21  ProdukModel(getUid(), waktuDibuat, namaProduk, kategoriProduk,
22                                  hargaProduk, satuanProduk,
23  namaSatuan, namaAreaPenjual, deskripsiProduk);
24                      produkPresenter.buatProdukBaru(produkModel,
25  listImage);
26                  }
27              } catch (Exception e) {
28                  ShowSnackbar.showSnack(this,
29  getResources().getString(R.string.msg_error));
30              }
31          } else {
32              final Snackbar snackbar = Snackbar.make(activity,
33  getResources().getString(R.string.msg_noInternet),
34  Snackbar.LENGTH_INDEFINITE);
35  snackbar.setAction(getResources().getString(R.string.ok), new
36  View.OnClickListener() {
37                  @Override
38                  public void onClick(View v) {
39                      snackbar.dismiss();
40                  }
41              });
42              snackbar.show();
43          }
44      } else {
45          ShowAlertDialog.showAlert("Anda harus mengisi semua Form
46  yang tersedia !", this);
47      }
48  }
```

Gambar 5.67 Implementasi Kode Program *Method* simpanProduk

Tabel 5.19 Penjelasan *Method* simpanProduk

Nomor Kode Program	Penjelasan
1	Inisialisasi nama *method* simpanProduk
2	Kondisi if untuk mengecek nilai method cekKolomIsian
3	Inisialisasi variabel namaProduk
4-5	Inisialisasi variabel hargaProduk
6-7	Inisialisasi variabel satuanProduk
8-9	Inisialisasi variabel deskripsiProduk
10	Inisialisasi variabel waktuDibuat dengan tipe data long dan nilai waktu

11-13	Kondisi if untuk mengecek status koneksi internet
14	Deklarasi statement try
15 – 17	Inisialisasi kondisi apabila variabel listImage memiliki nilai = 0, maka akan muncul dialog pesan
18	Deklarasi kondisi sebaliknya dari kondisi pada line 32-35
19	Memanggil method untuk menampilkan progresbar
20 – 23	Inisialisasi *class* produkModel dengan set nilai tertentu
24 – 25	Memanggil method buatProdukBaru dari *class* presenter dengan set nilai produkModel dan listImage
27	Deklarasi Statement catch untuk menampung pesan error
28 – 29	Menampilkan pesan error berupa snackbar
31 – 43	Kondisi sebaliknya dari nilai pada line 11 – 13, dan menampilkan pesan pada snackbar
44 – 47	Kondisi else dimana jika nilai dari *method* cekKolomIsian tidak sama dengan true maka akan menampilkan pesan dialog
48	Kurung penutup dari *method* simpanProduk

5.2.5.3 Implementasi *Method* buatProdukBaru

Method `buatProdukBaru` merupakan method yang berguna untuk melakukan submit nilai pada *firebase*. *Method* ini juga berfungsi memanggil *method* `uploadPhotoThreadListener` untuk melakukan proses upload foto secara *asynctas*k.

```
1   public void buatProdukBaru(final ProdukModel produkModel,
2   ArrayList<Uri> imageUri){
3       final String pushId =
4   mFirestore.collection(Constants.PRODUK_REGULER).document().getId
5   ();
6       String produkId = pushId;

7       HashMap<String, Object> dataProduk = new HashMap<>();
8       dataProduk.put(Constants.ID_PENJUAL, produkModel.getUid());
9       dataProduk.put(Constants.WAKTU_DIBUAT,
10  produkModel.getWaktuDibuat());
11      dataProduk.put(Constants.NAMAPRODUK,
12  produkModel.getNamaProduk());
13      dataProduk.put(Constants.ID_KATEGORI,
14  produkModel.getKategoriProduk());
15      dataProduk.put(Constants.HARGAPRODUK,
16  produkModel.getHargaProduk());
17      dataProduk.put(Constants.DIGITSATUAN,
18  produkModel.getSatuanProduk());
19      dataProduk.put(Constants.NAMASATUAN,
20  produkModel.getNamaSatuan());
21      dataProduk.put(Constants.ID_PRODUK, produkId);
22      dataProduk.put(Constants.GAMBARPRODUK,
```

```
23  produkModel.getGambarProduk());
24      dataProduk.put(Constants.ID_LOKASI,
25  produkModel.getIdLokasi());
26      dataProduk.put(Constants.DESKRIPSI,
27  produkModel.getDeskripsiProduk());

28  mFirestore.collection(Constants.PRODUK_REGULER).document(produkI
29  d).set(dataProduk).addOnSuccessListener(new
30  OnSuccessListener<Void>() {
31      @Override
32      public void onSuccess(Void aVoid) {
33          UploadPhotoThreadListener uploadPhotoThreadListener =
34  new UploadPhotoThreadListener() {
35              @Override
36              public void onUploadPhotoSuccess(ArrayList<String>
37  photoUrls) {
38                  Map<String, Object> gambarProduk = new
39  HashMap<>();
40                  gambarProduk.put(Constants.GAMBARPRODUK,
41  photoUrls);

42  mFirestore.collection(Constants.PRODUK_REGULER).document(pushId)
43  .update(gambarProduk);

44                  view.hideProgressDialog();
45                  view.finish();
46                  Toast.makeText(view.getApplicationContext(),
47  "Produk anda telah tersimpan", Toast.LENGTH_SHORT).show();
48              }
49          };
50          new UploadPhotoThread(pushId, imageUri,
51  uploadPhotoThreadListener).execute();
52      }
53  }).addOnFailureListener(new OnFailureListener() {
54      @Override
55      public void onFailure(@NonNull Exception e) {
56          Toast.makeText(view.getApplicationContext(),
57                  "Maaf, gagal menyimpan produk. Silahkan ulangi
58  kembali", Toast.LENGTH_SHORT).show();
59      }
60  });
61  }
```

Gambar 5.68 Implementasi Kode Program *Method* buatProdukBaru

Tabel 5.20 Penjelasan *Method* buatProdukBaru

Nomor Kode Program	Penjelasan
1 – 2	Inisialisasi nama *method* buatProdukBaru dengan nilai produkModel dan imageUri
3 – 5	Inisialisasi variabel pushId dengan tipe data string dan nilai dari push key pada database
6	Inisialisasi variabel produkId dengan tipe data string dan nilai dari variabel pushId

7	Inisialisasi variabel dataProduk dengan tipe data Hashmap
8 – 27	Set data nilai pada variabel dataProduk dengan beberapa atribut
28 – 32	Melakukan set value pada database produk_reguler, child idProduk dengan nilai dari dataProduk dan jika sukses maka akan memanggil interface UploadPhotoThreadListener
36 – 37	Inisialisasi *method* onUploadPhotoSucces
38 – 39	Inisialisasi variabel gambarProduk dengan tipe data HashMap
40 – 41	Set nilai variabel gambarProduk dengan nilai photoUrls
42 – 43	Melakukan update nilai pada database produk_reguler, child idProduk dengan nilai dari gambarProduk
44 – 45	Memanggil *method* hideProgessDialog dan finish dari view
46 – 47	Memanggil fungsi Toast dengan set nilai tertentu
50 – 51	Melakukan set nilai pada *method* UploadPhotoThread dan kemudian di eksekusi
53 - 60	Kondisi sebaliknya dari line 28 -32 dimana proses submit produk gagal, dan kemudian menampilkan pesan dalam Toast
61	Kurung penutup dari *method* buatProdukBaru

5.2.5.4 Implementasi *Method* buatProdukKhusus

Method `buatProdukKhusus` merupakan method yang berguna untuk melakukan submit nilai produk khusus pada *firebase* yang akan dipesan oleh konsumen. *Method* ini juga berfungsi memanggil *method* `pesanProdukKhusus`.

```
1   private void buatProdukKhusus() {
2           if (cekKolomIsian() == true) {
3               String namaProduk =
4   inputNamaProduk.getText().toString();
5               int satuanProduk =
6   Integer.parseInt(inputSatuanDigit.getText().toString());
7               if
8   (InternetConnection.getInstance().isOnline(PesanProdukKhususActi
9   vity.this)) {
10                  try {
11                      showProgessDialog();
12                      String pushId =
13  mDatabase.child(Constants.PRODUK_KHUSUS).child(kategoriProduk).p
14  ush().getKey();
15                      final String produkId = pushId;
16                      HashMap<String, Object> dataProduk = new
17  HashMap<>();
18                      dataProduk.put(Constants.NAMAPRODUK,
19  namaProduk);
20                      dataProduk.put(Constants.DIGITSATUAN,
21  satuanProduk);
```

```
22                      dataProduk.put(Constants.NAMASATUAN,
23  namaSatuan);
24                      dataProduk.put(Constants.ID_PRODUK,
25  produkId);
26  mFirestore.collection(Constants.PRODUK_KHUSUS).document(produkId
27  ).set(dataProduk).addOnSuccessListener(new
28  OnSuccessListener<Void>() {
29                          @Override
30                          public void onSuccess(Void aVoid) {
31                              pesanProdukKhusus(produkId);
32                          }
33                      }).addOnFailureListener(new
34  OnFailureListener() {
35                          @Override
36                          public void onFailure(@NonNull Exception
37  e) {
38  ShowSnackbar.showSnack(PesanProdukKhususActivity.this, "gagal
39  membuat produk pesanan, silahkan ulangi lagi");
40                          }
41                      });
42                  } catch (Exception e) {
43                      ShowSnackbar.showSnack(this,
44  getResources().getString(R.string.msg_error));
45                  }
46              } else {
47                  final Snackbar snackbar =
48  Snackbar.make(activity,
49  getResources().getString(R.string.msg_noInternet),
50  Snackbar.LENGTH_INDEFINITE);
51  snackbar.setAction(getResources().getString(R.string.ok), new
52  View.OnClickListener() {
53                      @Override
54                      public void onClick(View v) {
55                          snackbar.dismiss();
56                      }
57                  });
58                  snackbar.show();
59              }
60          } else {
61              ShowAlertDialog.showAlert("Anda harus mengisi semua
62  Form yang tersedia !", this);
63          }
64      }
```

Gambar 5.69 Implementasi Kode Program *Method* buatProdukKhusus

Tabel 5.21 Penjelasan *Method* buatProdukKhusus

Nomor Kode Program	Penjelasan
1	Inisialisasi nama *method* buatProdukKhusus
2	Kondisi if dimana jika nilai dari *method* cekKolomIsian sama dengan true eksekusi line berikutnya (line 3)

3 – 4	Inisialisasi variabel namaProduk dengan set nilai dari inputNamaProduk
5 - 6	Inisialisasi variabel satuanProduk dengan set nilai dari inputDatuanDigit
7 – 9	Kondisi if untuk mengecek status koneksi internet
10	Deklarasi statement try
11	Memanggil method untuk menampilkan progresbar
12 – 14	Inisialisasi variabel pushId dengan nilai key dari firebase
15	Inisialisasi variabel produkId dengan set nilai pushId
16 – 17	Inisialisasi variabel dataProduk dengan tipe HashMap
18 – 25	Melakukan set nilai pada variabel dataProduk dengan nilai dari masing masing konstanta
26 – 27	Menyimpan nilai dari variabel dataProduk pada database child ProdukKhusus, child produkId
27 – 32	Memanggil method addOnSuccesListener untuk mengetahui apakah produk berhasil dibuat, yang apabila succes akan memanggil *method* pesanProdukKhusus dengan set nilai produkId
33 – 41	Method sebaliknya dari kode baris 27 – 32, yang apabila gagal maka akan menampilkan pesan error dalam snackbar
42	Deklarasi statement catch untuk menampung pesan error
43 – 45	Menampilkan pesan error berupa snackbar
46 – 59	Kondisi sebaliknya dari nilai pada line 7 – 9, dan menampilkan pesan pada snackbar
60 – 63	Kondisi sebaliknya dari nilai pada line 5, dan menampilkan pesan pada snackbar
64	Kurung kurawal penutup *method* buatProdukKhusus

5.2.5.5 Implementasi *Method* pesanProdukKhusus()

Method `pesanProdukKhusus` merupakan method yang berguna untuk melakukan pemesanan produk secara khusus(tidak tersedia dalam daftar produk).

```
1   private void pesanProdukKhusus(String produkId) {
2       String idProduk = produkId;
3       String pesanKonsumen = notePenjual.getText().toString();
4       String tanggalKirim= inputTanggalKirim.getText().toString();
5       String waktuKirim = inputJamKirim.getText().toString();
6       long orderTime = new Date().getTime();

7       if
8   (InternetConnection.getInstance().isOnline(PesanProdukKhususActi
9   vity.this)) {
```

```
10          try {
11              String pushId =
12  mDatabase.child(Constants.KONSUMEN).child(getUid()).child(Consta
13  nts.DAFTAR_TRANSAKSI).push().getKey();
14              String transaksiId = pushId;
15              Map<String, Object> pemesanan = new HashMap<>();
16              pemesanan.put(Constants.BIAYA_KIRIM, 0);
17              pemesanan.put(Constants.NOTE_KONSUMEN,
18  pesanKonsumen);
19              pemesanan.put(Constants.ID_KONSUMEN, getUid());
20              pemesanan.put(Constants.ID_TRANSAKSI, transaksiId);
21              pemesanan.put(Constants.ID_PENJUAL, penjualId);
22              pemesanan.put(Constants.ID_PRODUK, idProduk);
23              pemesanan.put(Constants.JUMLAH_ORDER_PRODUK, 1);
24              pemesanan.put(Constants.NAMA_PENERIMA, " ");
25              pemesanan.put(Constants.STATUS_TRANSAKSI, 1);
26              pemesanan.put(Constants.JENIS_PRODUK,
27  Constants.PRODUK_KHUSUS);
28              pemesanan.put(Constants.JUMLAH_HARGA_PRODUK, 0);
29              pemesanan.put(Constants.TANGGAL, orderTime);
30              pemesanan.put(Constants.TANGGAL_KIRIM,
31  tanggalKirim);
32              pemesanan.put(Constants.WAKTU_KIRIM, waktuKirim);

33  mDatabase.child(Constants.KONSUMEN).child(getUid()).child(Consta
34  nts.DAFTAR_TRANSAKSI).child(Constants.PEMBELIAN_BARU).child(tran
35  saksiId).setValue(pemesanan);

36  mDatabase.child(Constants.PENJUAL).child(penjualId).child(Consta
37  nts.DAFTAR_TRANSAKSI).child(Constants.PENJUALAN_BARU).child(tran
38  saksiId).setValue(pemesanan)
39                          .addOnSuccessListener(new
40  OnSuccessListener<Void>() {
41                              @Override
42                              public void onSuccess(Void aVoid) {
43                                  buatNotifikasiOrder();
44                                  finish();

45  Toast.makeText(getApplicationContext(), "Anda telah berhasil
46  melakukan pemesanan produk", Toast.LENGTH_SHORT).show();
47                              }
48                          }).addOnFailureListener(new
49  OnFailureListener() {
50                      @Override
51                      public void onFailure(@NonNull Exception e) {
52  ShowSnackbar.showSnack(PesanProdukKhususActivity.this, "gagal
53  membuat pesanan, silahkan ulangi lagi");
54                      }
55              });
56          } catch (Exception e) {
57              ShowSnackbar.showSnack(this,
58  getResources().getString(R.string.msg_error));
59          }
60      } else {
61          final Snackbar snackbar = Snackbar.make(activity,
62  getResources().getString(R.string.msg_noInternet),
63  Snackbar.LENGTH_INDEFINITE);

64  snackbar.setAction(getResources().getString(R.string.ok), new
```

```
65  View.OnClickListener() {
66              @Override
67              public void onClick(View v) {
68                  snackbar.dismiss();
69              }
70          });
71          snackbar.show();
72      }
73  }
```

Gambar 5.70 Implementasi Kode Program *Method* pesanProdukKhusus

Tabel 5.22 Penjelasan *Method* pesanProdukKhusus

Nomor Kode Program	Penjelasan
1	Inisialisasi nama *method* pesanProdukKhusus dengan nilai produkId
2	Inisialisasi variabel idProduk dengan set nilai produkId
3	Inisialisasi variabel pesanKonsumen dengan set nilai dari notePenjual
4	Inisialisasi variabel tanggalKirim dengan set nilai dari inputTanggalKirim
5	Inisialisasi variabel waktuKirim dengan set nilai dari inputJamKirim
6	Inisialisasi variabel orderTime dengan set nilai dari fungsi Date
7 – 9	Kondisi if untuk mengecek status koneksi internet
10	Deklarasi statement try
11 – 13	Inisialisasi variabel produkId dengan set nilai pushKey dari firebase
14	Inisialisasi variabel pemesananId dengan variabel pushId
15	Inisialisasi variabel pemesanan dengan tipe HashMap
16 – 32	Melakukan set nilai pada variabel pemesanan dengan nilai dari masing masing atribut
33 – 35	Menyimpan nilai dari variabel pemesanan pada database Konsumen, child id konsumen, child daftarTransaksi, child pembelianBaru, child transaksiId
36 – 38	Menyimpan nilai dari variabel pemesanan pada database Konsumen, child id penjual, child daftarTransaksi, child pejualanBaru, child transaksiId
39 – 44	Memanggil method addOnSuccesListener untuk mengetahui apakah produk berhasil dibuat, yang apabila succes akan memanggil *method* buatNotifikasiOrder
45 – 46	Memanggil fungsi Toast dengan set nilai tertentu

48 – 55	Method sebaliknya dari kode baris 39 – 44, yang apabila gagal maka akan menampilkan pesan error dalam snackbar
56	Deklarasi statement catch untuk menampung pesan error
57 – 58	Menampilkan pesan error berupa snackbar
60 – 72	Kondisi sebaliknya dari nilai pada line 7 – 9, dan menampilkan pesan pada snackbar
73	Kurung kurawal penutup *method* pesanProdukKhusus

5.2.6 Implementasi Antarmuka

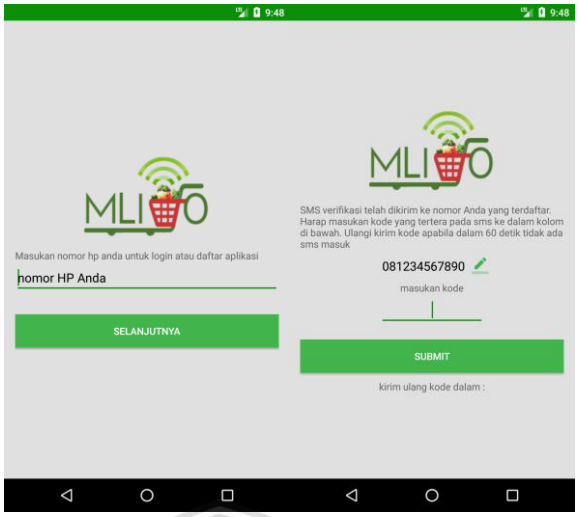
Implementasi antarmuka menjelaskan bagaimana hasil tampilan aplikasi MLIJO yang telah dikembangkan berdasarkan fungsi sistem, perancangan antarmuka, dan juga hasil dari iterasi 1 pengguna. Implementasi antarmuka perangkat lunak Mlijo terdiri dari 3 bagian, yaitu implementasi antarmuka pengguna, antarmuka konsumen dan antarmuka penjual.

5.2.6.1 Implementasi Antarmuka Aplikasi Pengguna

Antarmuka pada sisi pengguna berupa halaman yang disediakan bagi pengguna untuk dapat masuk kedalam sistem. Antarmuka untuk sisi pengguna terdiri dari antarmuka halaman autentifikasi dan input data user baru. Hasil implementasi pada sisi pengguna ini berbeda dengan antarmuka pada perancangan di karenakan saat iterasi 0 penulis melakukan perubahan fungsi. Perubahan yang dilakukan adalah alur dan data input autentifikasi yang semula menggunakan email dan *password* menjadi mengguna nomor ponsel. Alasan perubahan tersebut adalah dikarenakan kini nomor ponsel dianggap lebih aman dalam memverifikasi kebenaran pengguna, sebab nomor ponsel saat ini telah diverifikasi oleh pemerintah sesuai dengan NIK penduduk Indonesia. Antarmuka pengguna terdiri halaman autentifikasi, input data user baru konsumen, dan input data user baru penjual.

a. Halaman Autentifikasi

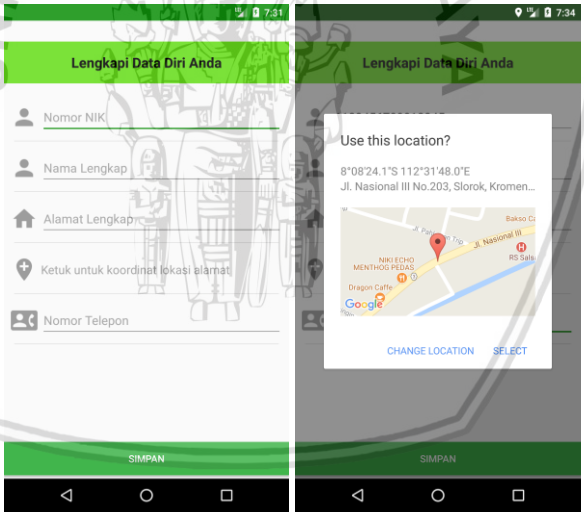
   Halaman Autentifikasi merupakan salah satu antarmuka pengguna pada sisi aplikasi konsumen dan penjual yang berfungsi untuk melakukan login atau *register* ke dalam sistem. Halaman antarmuka autentifikasi ini terbagi menjadi dua, yaitu halaman input nomor ponsel dan halaman input kode verifikasi. Antarmuka autentifikasi ditunjukan oleh gambar 5.71 berikut :



Gambar 5.71 Implementasi Antarmuka Halaman Autentifikasi

b. Halaman Input Data User Baru Konsumen

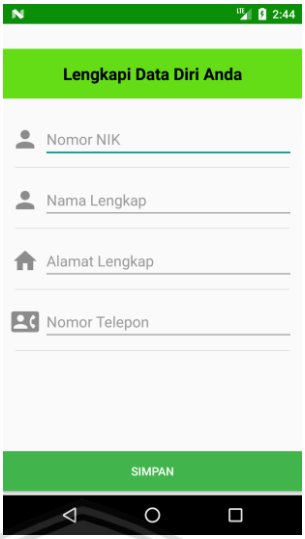
Halaman Input Data User Baru Konsumen merupakan salah satu antarmuka pengguna pada sisi aplikasi konsumen yang berfungsi untuk melengkapi data diri konsumen yang baru mendaftar. Antarmuka Input Data User Baru Konsumen ditunjukan oleh gambar 5.72 berikut :



Gambar 5.72 Implementasi Antarmuka Halaman Input Data User Baru Konsumen

c. Halaman Input Data user Baru Penjual

Halaman Input Data User Baru Penjual merupakan salah satu antarmuka pengguna pada sisi aplikasi penjual yang berfungsi untuk melengkapi data diri penjual yang baru mendaftar. Antarmuka Input Data User Baru Penjual ditunjukan oleh gambar 5.73 berikut :



Gambar 5.73 Implementasi Antarmuka Halaman Input Data User Baru Penjual

5.2.6.2 Implementasi Antarmuka Aplikasi Konsumen

Antarmuka pada sisi konsumen berupa halaman yang disediakan bagi konsumen untuk dapat menggunakan semua fungsionalitas sistem. Hasil implementasi pada sisi konsumen ini secara umum seperti pada antarmuka perancangan, namun terdapat sedikit perubahan tampilan dan juga fungsi. Selain itu juga terdapat fungsi yang dihapus yaitu fungsi keranjang belanja karena perubahan alur sistem yang mana tidak memerlukan fungsi keranjang belanja, kemudian info gizi karena tidak mendapat feedback baik dari konsumen dan kelola alamat. Antarmuka untuk sisi konsumen terdiri dari antarmuka halaman *dashboard*, pencarian, produk, penjual, info lokasi penjual, obrolan, kelola pembelian, ulasan, info harga, pengaturan dan halaman lainnya.

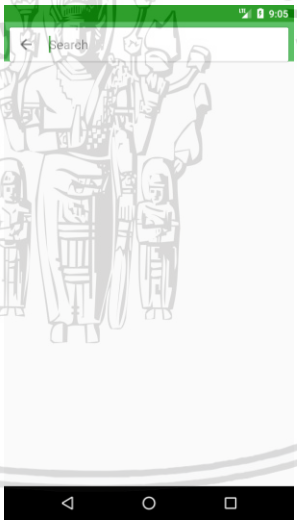
a. Halaman *Dashboard* Produk

   Halaman *dashboard* produk merupakan salah satu antarmuka konsumen yang merupakan halaman utaman konsumen. Halaman *dashboard* produk berfungsi untuk melakukan pencarian produk berdasarkan daftar kategori yang tersedia atau dengan kata kunci pada halaman pencarian yang dapat di akses oleh tombol FAB. Antarmuka *dashboard* produk ditunjukan oleh gambar 5.74 berikut :

Gambar 5.74 Implementasi Antarmuka Halaman *Dashboard* Produk

b. Halaman Input Pencarian

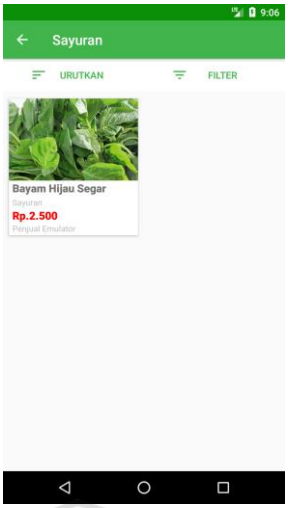
Halaman Input Pencarian merupakan salah satu antarmuka konsumen yang berfungsi untuk melakukan pencarian produk berdasarkan kata kunci. Antarmuka pencarian ditunjukan oleh gambar 5.75 berikut :



Gambar 5.75 Implementasi Antarmuka Halaman Pencarian

c. Halaman Daftar Produk

Halaman daftar produk merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan list daftar produk sesuai dengan kategori terpilih maupun pencarian kata kunci. Antarmuka daftar produk ditunjukan oleh gambar 5.76 berikut :



Gambar 5.76 Implementasi Antarmuka Halaman Daftar Produk

d. Halaman Detail Produk

Halaman Detail Produk merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan detail dari sebuah produk. Pada antarmuka detail produk ini terdapat sedikit perubahan dimana terjadi perubahan tata letak keterangan produk dan juga penambahan informasi penjual sesuai dengan hasil iterasi 1. Antarmuka detail produk ditunjukan oleh gambar 5.77 berikut :

Gambar 5.77 Implementasi Antarmuka Halaman Detail Produk

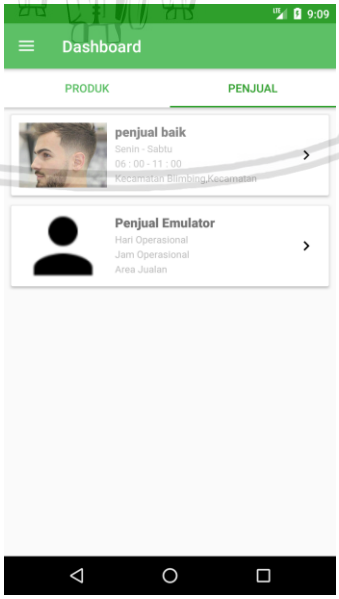
e. Halaman Pesan Produk

Halaman Pesan Produk merupakan salah satu antarmuka konsumen yang berfungsi untuk melakukan pemesanan produk. Pada antarmuka pesan produk ini perubahan berupa penambahan fungsi input jumlah produk yang sebelumnya pada halaman detail produk. Kemudian penambahan fungsi input waktu pengiriman pesanan. Selain itu, fungsi pilih dan tambah alamat dihapus. Antarmuka pesan produk ditunjukan oleh gambar 5.78 :

Gambar 5.78 Implementasi Antarmuka Halaman Pesan Produk

f. Halaman *Dashboard* Penjual

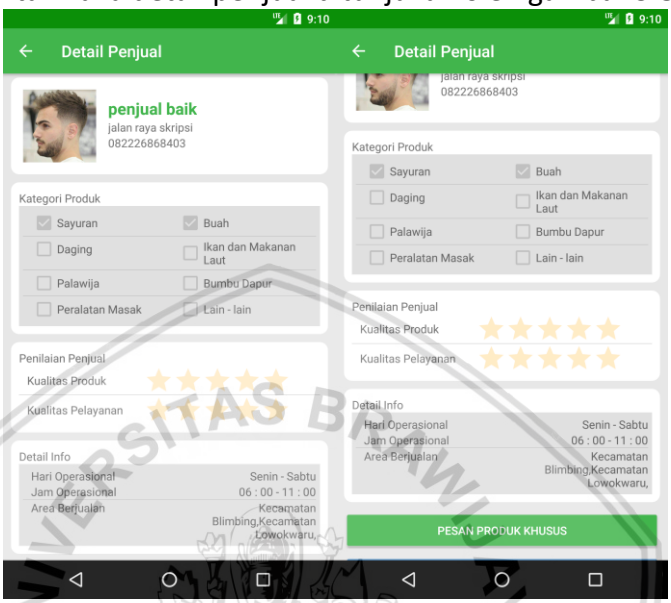
Halaman *dashboard* penjual merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan daftar penjual terbaru. Antarmuka tersebut merupakan pergabungan dari perancangan antarmuka *Dashboard* penjual dan Daftar penjual. Perubahan tersebut dilakukan oleh penulis dengan pertimbangan agar lebih efisien disamping karena fungsi image slider pada sisi penjual kurang dibutuhkan secara fungsional. Antarmuka *dashboard* penjual ditunjukan oleh gambar 5.79 berikut :



Gambar 5.79 Implementasi Antarmuka Halaman *Dashboard* Penjual

g. Halaman Detail Penjual

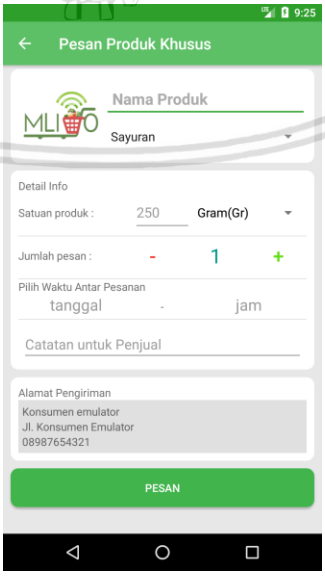
Halaman detail penjual merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan detail dari profil penjual. Selain itu pada halaman tersebut terdapat tombol untuk menuju halaman pesan produk khusus, halaman kirim obrolan, dan halaman daftar produk dari penjual tersebut. Antarmuka detail penjual ditunjukan oleh gambar 5.80 berikut :



Gambar 5.80 Implementasi Antarmuka Halaman Detail Penjual

h. Halaman Pesan Produk Khusus

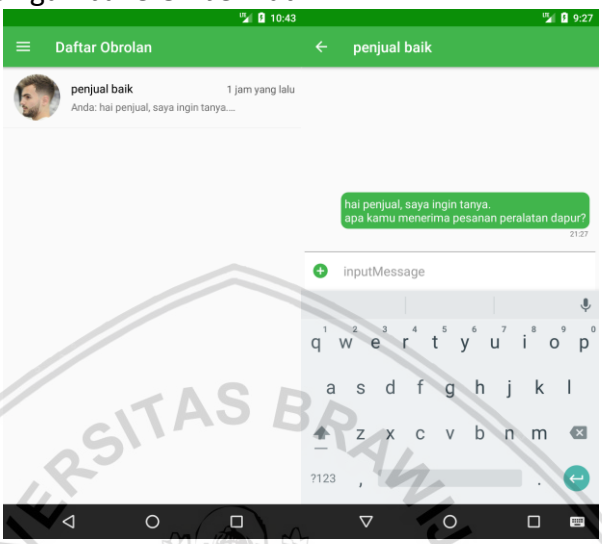
Halaman pesan produk khusus merupakan salah satu antarmuka konsumen yang berfungsi untuk melakukan pemesanan produk secara khusus terhadap penjual tertentu. Antarmuka pesan produk khusus ditunjukan oleh gambar 5.81 berikut :



Gambar 5.81 Implementasi Antarmuka Halaman Pesan Produk Khusus

i. Halaman Kirim Obrolan

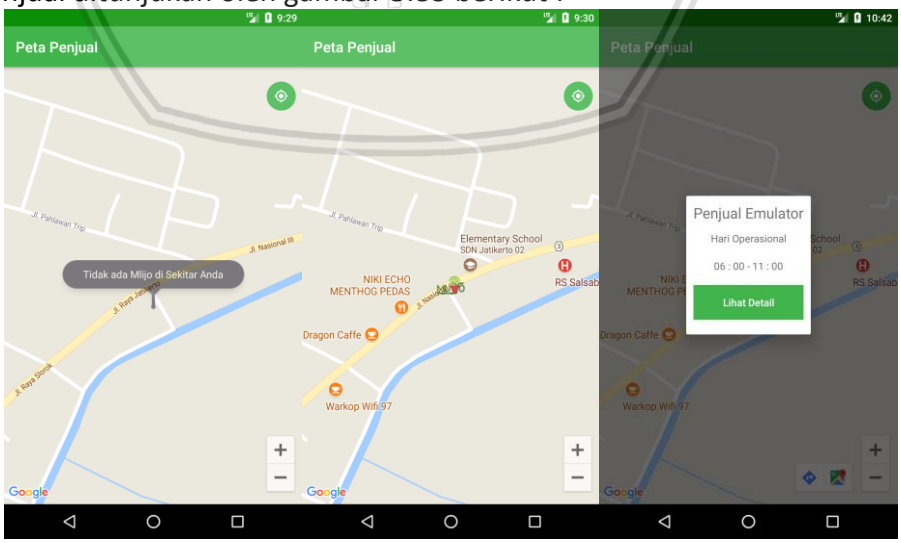
Halaman kirim obrolan merupakan salah satu antarmuka konsumen yang berfungsi untuk melakukan kirim pesan dengan penjual tertentu. Fungsi ini juga ditambahkan fitur untuk kirim gambar sesuai hasil iterasi 1. Fungsi dan antarmuka ini juga terdapat pada sisi penjual. Antarmuka kirim obrolan ditunjukan oleh gambar 5.82 berikut :



Gambar 5.82 Implementasi Antarmuka Halaman Kirim Obrolan

j. Halaman Peta Penjual Aktif

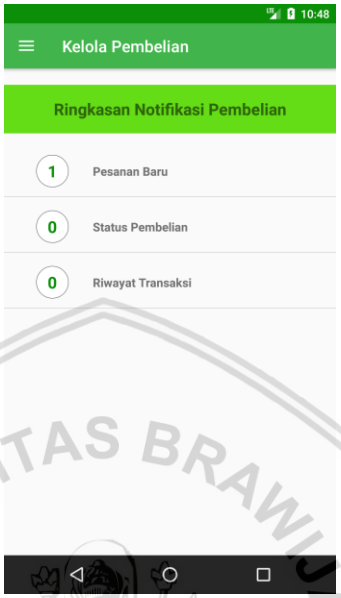
Halaman peta penjual merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan peta lokasi penjual aktif. Pada fitur tersebut dilakukan penambahan fungsi onClickMarker dimana saat marker penjual di klik akan muncul dialog builder informasi penjual. Pada builder tersebut terdapat tombol untuk menuju kehalaman detail penjual. Antarmuka peta penjual ditunjukan oleh gambar 5.83 berikut :



Gambar 5.83 Implementasi Antarmuka Halaman Peta Penjual Aktif

k. Halaman Daftar Kelola Pembelian

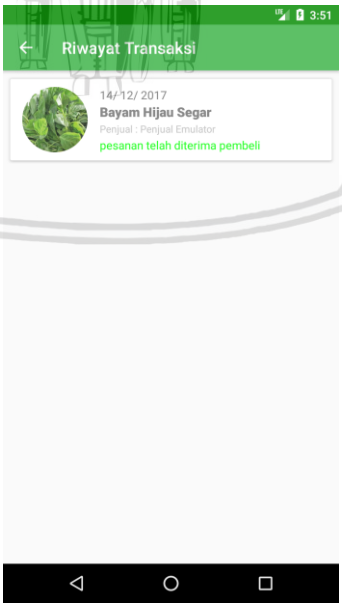
Halaman daftar kelola pembelian merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan daftar menu pesanan baru, status pesanan dan riwayat pesanan. Antarmuka daftar kelola pembelian ditunjukan oleh gambar 5.84 berikut :



Gambar 5.84 Implementasi Antarmuka Halaman Kelola Pembelian

l. Halaman Daftar Transaksi

Halaman daftar transaksi merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan list transaksi pembelian. Antarmuka daftar transaksi ditunjukan oleh gambar 5.85 berikut :



Gambar 5.85 Implementasi Antarmuka Halaman Daftar Transaksi

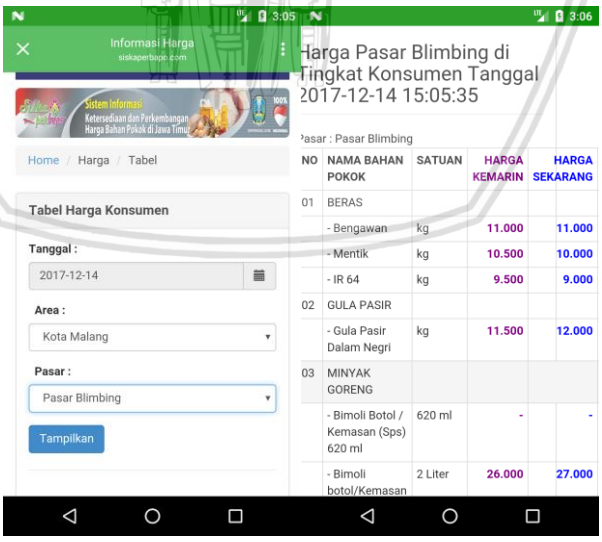
m. Halaman Detail Transaksi

Halaman detail transaksi merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilan detail dari transaksi. Antarmuka detail transaksi ditunjukan oleh gambar 5.86 berikut :

Gambar 5.86 Implementasi Antarmuka Halaman Detail Transaksi

n. Halaman Info Harga

Halaman Info harga merupakan salah satu antarmuka konsumen yang berfungsi untuk menampilkan tabel informasi harga terbaru setiap harinya. Antarmuka info harga ditunjukan oleh gambar 5.87 berikut :



Gambar 5.87 Implementasi Antarmuka Halaman Info Harga

5.2.6.3 Implementasi Antarmuka Aplikasi Penjual

Antarmuka pada sisi penjual berupa halaman yang disediakan bagi konsumen untuk dapat menggunakan semua fungsionalitas sistem. Hasil implementasi pada

sisi penjual ini secara umum seperti pada antarmuka perancangan, namun terdapat sedikit perubahan tampilan dan juga fungsi. Antarmuka untuk sisi penjual terdiri dari antarmuka halaman *dashboard*, produk, kelola penjualan, peta lokasi konsumen, obrolan, ulasan, info harga, pengaturan dan halaman lainnya

a. Halaman *Dashboard*

Halaman *dashboard* merupakan salah satu antarmuka penjual yang berfungsi untuk menampilkan status berjualan penjual dan tombol switch untuk merubah status tersebut. Antarmuka *dashboard* ditunjukan oleh gambar 5.88 berikut :

Gambar 5.88 Implementasi Antarmuka Halaman *Dashboard*

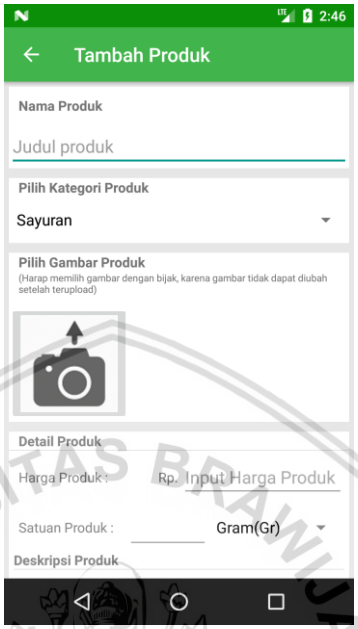
b. Halaman Kelola Produk

Halaman kelola produk merupakan salah satu antarmuka penjual yang berfungsi untuk menampilkan daftar produk beserta menu tambah, ubah dan hapus produk. Antarmuka kelola produk ditunjukan oleh gambar 5.89:

Gambar 5.89 Implementasi Antarmuka Halaman Kelola Produk

c. Halaman Tambah Produk

Halaman tambah produk merupakan salah satu antarmuka penjual yang berfungsi untuk membuat data produk baru. Antarmuka tambah produk ditunjukan oleh gambar 5.90 berikut :



Gambar 5.90 Implementasi Antarmuka Halaman Tambah Produk

d. Halaman Detail Produk

Halaman detail produk merupakan salah satu antarmuka penjual yang berfungsi untuk menampilkan detail data dari sebuah produk. Antarmuka detail produk ditunjukan oleh gambar 5.91 berikut :

Gambar 5.91 Implementasi Antarmuka Halaman Detail Produk

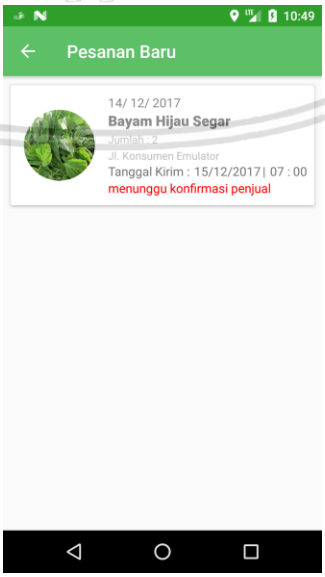
e. Halaman Ubah Produk

Halaman ubah produk merupakan salah satu antarmuka penjual yang berfungsi untuk melakukan perubahan data pada produk tertentu. Pada antarmuka ini terdapat perubahan dimana penjual tidak dapat melakukan ubah kategori dan gambar produk. Antarmuka ubah produk ditunjukan oleh gambar 5.92 berikut :

Gambar 5.92 Implementasi Antarmuka Halaman Ubah Produk

f. Halaman Daftar Transaksi

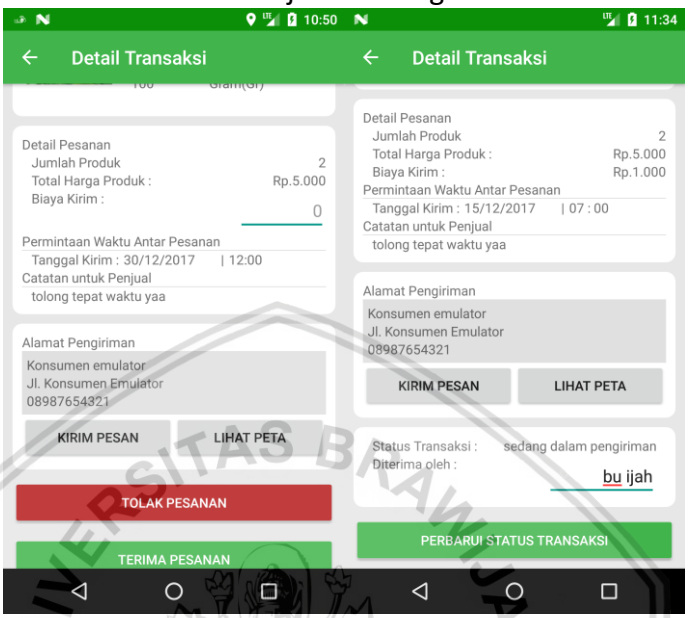
Halaman daftar transaksi merupakan salah satu antarmuka penjual yang berfungsi untuk menampilkan daftar transaksi penjualan. Antarmuka daftar transaksi ditunjukan oleh gambar 5.93 berikut :



Gambar 5.93 Implementasi Antarmuka Halaman Daftar Transaksi

g. Halaman Detail Transaksi

Halaman detail transaksi merupakan salah satu antarmuka penjual yang berfungsi untuk menampilan detail dari transaksi dan juga terdapat tombol fungsi untuk menerima, menolak dan memperbarui status transaksi. Antarmuka detail transaksi ditunjukan oleh gambar 5.94 berikut :



Gambar 5.94 Implementasi Antarmuka Halaman Detail Transaksi

h. Halaman Peta Konsumen

Halaman peta konsumen merupakan salah satu antarmuka penjual yang berfungsi untuk manampilkan peta lokasi konsumen yang melakukan pemesanan. Antarmuka peta konsumen ditunjukan oleh gambar 5.95 berikut:

Gambar 5.95 Implementasi Antarmuka Halaman Peta Konsumen

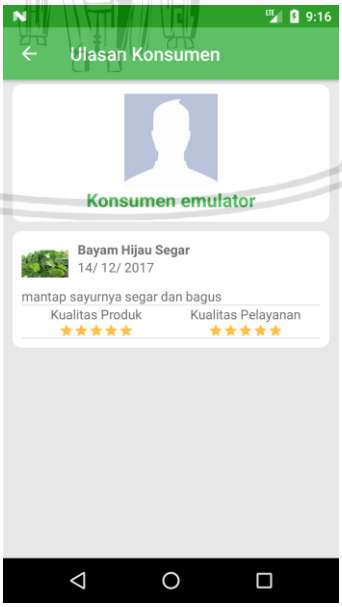
i. Halaman Daftar Ulasan

Halaman daftar ulasan merupakan salah satu antarmuka penjual yang berfungsi untuk menampilkan daftar ulasan yang pernah konsumen berikan kepada penjual. Antarmuka beri ulasan ditunjukan oleh gambar 5.96 berikut :

Gambar 5.96 Implementasi Antarmuka Halaman Daftar Ulasan

j. Halaman Detail Ulasan

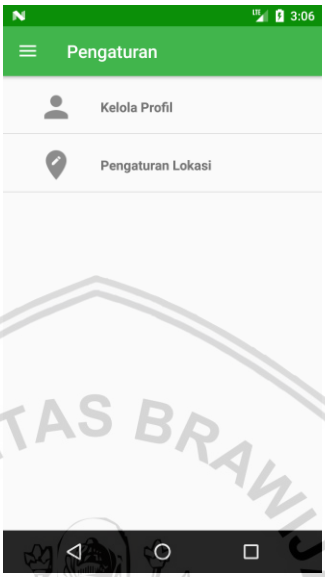
Halaman detail ulasan merupakan salah satu antarmuka penjual yang berfungsi untuk menampilkan detail dari sebuah ulasan. Antarmuka detail ulasan ditunjukan oleh gambar 5.97 berikut :



Gambar 5.97 Implementasi Antarmuka Halaman Detail Ulasan

k. Halaman Pengaturan

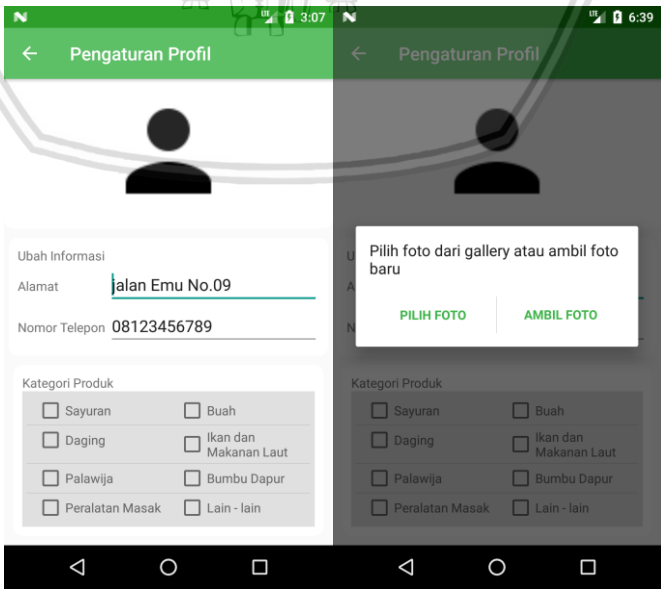
Halaman pengaturan merupakan salah satu antarmuka penjual yang berfungsi untuk menampilkan menu pengaturan profil dan menu pengaturan lokasi. Antarmuka pengaturan ditunjukan oleh gambar 5.98 berikut :



Gambar 5.98 Implementasi Antarmuka Halaman Pengaturan

l. Halaman Pengaturan Profil

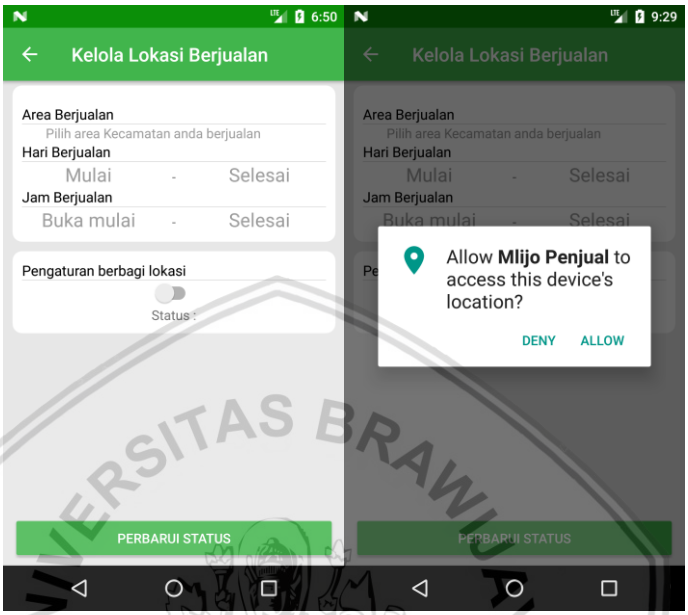
Halaman pengaturan profil merupakan salah satu antarmuka penjual yang berfungsi untuk melakukan upload foto profil, ubah alamat, ubah nomor telepon, dan data kategori produk yang dijual. Antarmuka pengaturan profil ditunjukan oleh gambar 5.99 berikut :



Gambar 5.99 Implementasi Antarmuka Halaman Pengaturan Profil

m. Halaman Pengaturan Lokasi

Halaman pengaturan lokasi merupakan salah satu antarmuka penjual yang berfungsi untuk mengatur info area berjualan penjual, waktu berjualan dan juga fitur berbagi lokasi. Antarmuka pengaturan lokasi ditunjukan oleh gambar 5.100 berikut :



Gambar 5.100 Implementasi Antarmuka Halaman Pengaturan Lokasi

BAB 6 PENGUJIAN

Bab ini menjelaskan mengenai tahapan pengujian perangkat lunak MLIJO yang telah di kembangkan sebelumnya. Pengujian pada aplikasi MLIJO ini mengadopsi fase *System Test & Fix* dari metode *Mobile*-D. Dimana pada fase tersebut menggunakan teknik *Test Driven Development*(TDD). Teknik TDD memiliki 2 *level* yaitu *Acceptance Test Driven Development*(ATDD) dan *Developer Test Driven Development*(*Developer TDD*). ATDD atau *Acceptence Test* sendiri umum di kenal sebagai *blackbox testing*, sedangkan *Developer TDD* juga dikenal sebagai *whitebox testing* (Ambler, 2013).

Proses pengujian dilakukan melalui tiga tahapan (strategi) yaitu pengujian unit, pengujian integrasi, dan pengujian validasi. Pada pengujian unit dan integrasi, akan digunakan teknik pengujian *Developer TDD (White-Box Testing)*. Pada pengujian validasi akan digunakan teknik pengujian ATDD (*Black-Box Testing*). Selain itu pada fase ini juga terdapat aktivitas *testing* yang terdiri dari *task acceptance test*. Pada bagian ini akan dilakukan pengujian *acceptance* dengan parameter u*sability* yang berhubungan dengan kebutuhan non-fungsional sistem.

6.1 Pengujian Fungsional

Pengujian Fungsional bertujuan untuk menguji seluruh kebutuhan fungsional yang telah dikembangkan dalam sistem. Dalam pengujian fungsional terbagi menjadi pengujian unit, pengujian integrasi, pengujian validasi dan pengujian *acceptance*. Pada tahap ini pengujian unit dan integrasi hanya di lakukan pada *method – method* dari fungsi utama sistem saja.

6.1.1 Test Driven Development

Test driven development bertujuan untuk melakukan pengujian terhadap alur logika algoritme yang telah di rancang sebelumnya sebelum impelementasi kode. Hal tersebut dilakukan untuk membuktikan bahwa alur logika yang dirancang telah berjalan dengan semestinya dan tidak salah. Pada tdd ini menghasilkan beberapa hal yaitu berupa penentuan skenario uji, implementasi kode pengujian dan hasil pengujian dari setiap skenario uji.

6.1.1.1 Pengujian algoritme CekKolomIsian

Pengujian algoritme cekKolomIsian dilakukan untuk menguji alur logic method cekKolomIsian sebelum di implementasikan kedalam kode program. Pada pengujian ini terdapat dua skenario, dimana skenario pertama mengecek apakah kolom isian tidak kosong atau kondisi salah, dan skenario kedua mengecek apakah kolom isian kosong atau kondisi benar.

a. Skenario pengujian

Tabel 6.1 Skenario pengujian cekKolomIsian kondisi salah

Nama Kasus Uji	Cek kolom isian kondisi kolom tidak kosong

Tujuan Pengujian	Untuk membuktikan bahwa kolom input kosong atau tidak
Prosedur Pengujian	1. inisialisasi nama kolom 2. melakukn cek apakah tidak terdapat kolom yang kosong
Hasil yang Diharapkan	Sistem menampilkan pesan salah

Tabel 6.2 Skenario pengujian cekKolomIsian kondisi benar

Nama Kasus Uji	Cek kolom isian kondisi kolom kosong
Tujuan Pengujian	Untuk membuktikan bahwa kolom input kosong atau tidak
Prosedur Pengujian	1. inisialisasi nama kolom 2. melakukn cek apakah terdapat kolom yang kosong
Hasil yang Diharapkan	Sistem menampilkan pesan benar

b. Implementasi skenario pengujian

Tabel 6.3 Implementasi pengujian cekKolomIsian kondisi salah

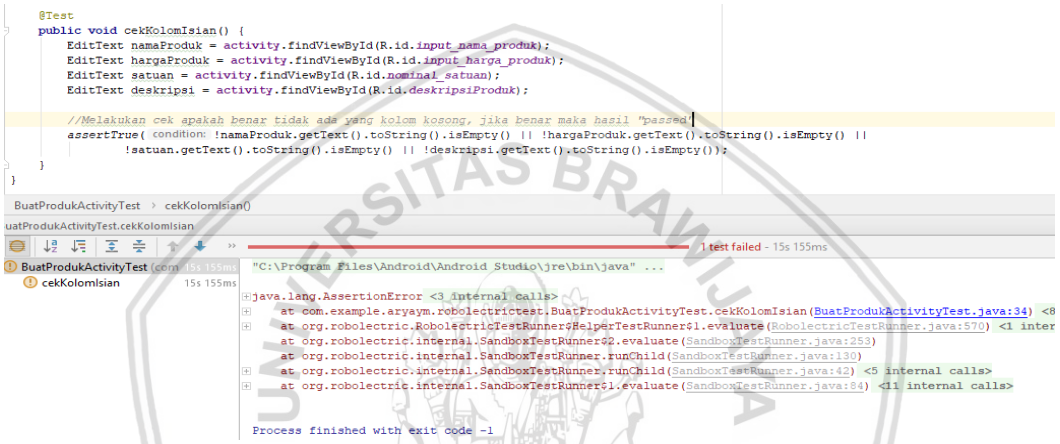
```
1  @Test
2  public void cekKolomIsian() {
3      EditText namaProduk =
4  activity.findViewById(R.id.input_nama_produk);
5      EditText hargaProduk =
6  activity.findViewById(R.id.input_harga_produk);
7      EditText satuan =
8  activity.findViewById(R.id.nominal_satuan);
9      EditText deskripsi =
10 activity.findViewById(R.id.deskripsiProduk);
11
12     //Melakukan cek apakah benar tidak ada kolom kosong,
13 jika benar maka hasil "passed"
14     assertTrue(!namaProduk.getText().toString().isEmpty()
15 || !hargaProduk.getText().toString().isEmpty() ||
16             !satuan.getText().toString().isEmpty() ||
17 !deskripsi.getText().toString().isEmpty());
18 }
```

Tabel 6.4 Implementasi pengujian cekKolomIsian kondisi benar

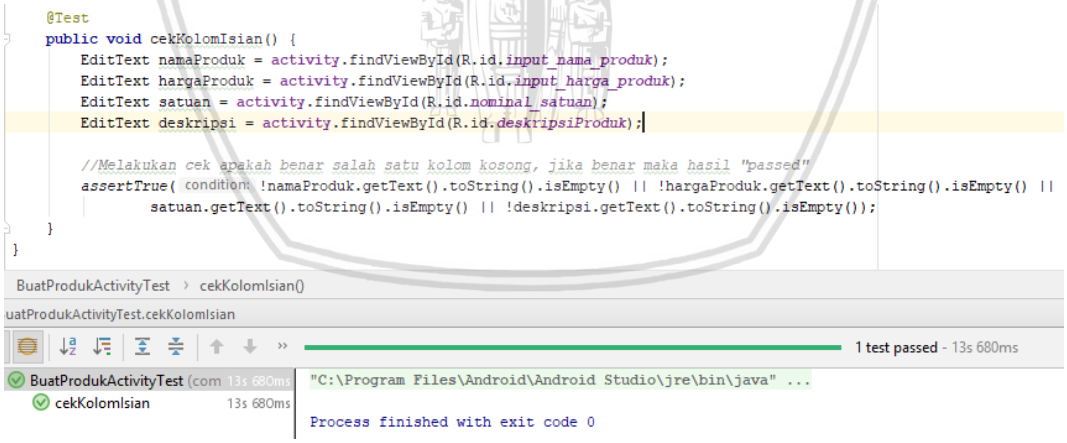
```
1  @Test
2  public void cekKolomIsian() {
3      EditText namaProduk =
4  activity.findViewById(R.id.input_nama_produk);
5      EditText hargaProduk =
6  activity.findViewById(R.id.input_harga_produk);
7      EditText satuan =
```

```
 8  activity.findViewById(R.id.nominal_satuan);
 9      EditText deskripsi =
10  activity.findViewById(R.id.deskripsiProduk);
11
12      //Melakukan cek apakah benar salah satu kolom kosong,
13  jika benar maka hasil "passed"
14      assertTrue(namaProduk.getText().toString().isEmpty() ||
15  hargaProduk.getText().toString().isEmpty() ||
16              satuan.getText().toString().isEmpty() ||
17  deskripsi.getText().toString().isEmpty());
18  }
```

c. Hasil pengujian



Gambar 6.1 Hasil pengujian cekKolomIsian kondisi salah



Gambar 6.2 Hasil pengujian cekKolomIsian kondisi benar

6.1.1.2 Pengujian Algoritme pesanProdukKhusus

Pengujian algoritme pesanProdukKhusus dilakukan untuk menguji alur logic method pesanProdukKhusus sebelum di implementasikan kedalam kode program. Pada pengujian ini terdapat empat skenario, dimana skenario pertama mengecek apakah semua nilai bernilai true atau kondisi benar, skenario kedua mengecek apabila simpan produk gagal, kondisi ketiga mengecek apabila kondisi terdapat

kesalahan sistem dan kondisi empat mengecek apabila kondisi tidak terdapat koneksi internet.

a. Skenario pengujian

Tabel 6.5 Skenario pengujian pesanProdukKhusus kondisi benar

Nama Kasus Uji	Pesan produk khusus kondisi semua benar
Tujuan Pengujian	Untuk membuktikan bahwa jika kondisi koneksi internet, simpan data bernilai benar dan tidak terdapat kesalahan sistem maka hasil uji benar
Prosedur Pengujian	1. inisialisasi nilai variabel 2. melakukan pembuktian terhadap variabel apakah bernilai benar
Hasil yang Diharapkan	Sistem menampilkan pengujian bernilai benar

Tabel 6.6 Skenario pengujian pesanProdukKhusus kondisi simpan data salah

Nama Kasus Uji	Pesan produk khusus kondisi simpan data salah
Tujuan Pengujian	Untuk membuktikan bahwa jika kondisi koneksi internet benar dan simpan data salah
Prosedur Pengujian	1. inisialisasi nilai variabel 2. melakukan pembuktian terhadap variabel apakah bernilai benar
Hasil yang Diharapkan	Sistem menampilkan pengujian bernilai salah

Tabel 6.7 Skenario pengujian pesanProdukKhusus kondisi terjadi kesalahan sistem

Nama Kasus Uji	Pesan produk khusus kondisi semua benar
Tujuan Pengujian	Untuk membuktikan bahwa jika kondisi terdapat kesalahan sistem maka hasil uji salah
Prosedur Pengujian	1. Inisialisasi test exception 2. inisialisasi nilai variabel 3. melakukan pembuktian terhadap variabel apakah bernilai benar
Hasil yang Diharapkan	Sistem menampilkan pengujian bernilai salah

Tabel 6.8 Skenario pengujian pesanProdukKhusus kondisi tidak terdapat koneksi internet

Nama Kasus Uji	Pesan produk khusus kondisi semua benar
Tujuan Pengujian	Untuk membuktikan bahwa jika kondisi tidak terdapat koneksi internet
Prosedur Pengujian	1. inisialisasi nilai variabel 2. melakukan pembuktian terhadap variabel apakah bernilai benar
Hasil yang Diharapkan	Sistem menampilkan pengujian bernilai salah

b. Implementasi skenario pengujian

Tabel 6.9 Implementasi pengujian pesanProdukKhusus kondisi benar

```
1  @Test
2  public void pesanProdukKhusus() throws Exception{
3      boolean connetionIsOnline = true, simpanDataSukses = true;
4
5      assertTrue("terdapat koneksi internet dan simpan data
6  sukses", connetionIsOnline && simpanDataSukses);
7  }
8
```

Tabel 6.10 Implementasi pengujian pesanProdukKhusus kondisi simpan data salah

```
1  @Test
2  public void pesanProdukKhusus() throws Exception{
3      boolean connetionIsOnline = true, simpanDataSukses =
4  false;
5
6      assertTrue("terdapat koneksi internet dan simpan data
7  gagal", connetionIsOnline && simpanDataSukses);
8  }
```

Tabel 6.11 Implementasi pengujian pesanProdukKhusus kondisi terdapat kesalahan sistem

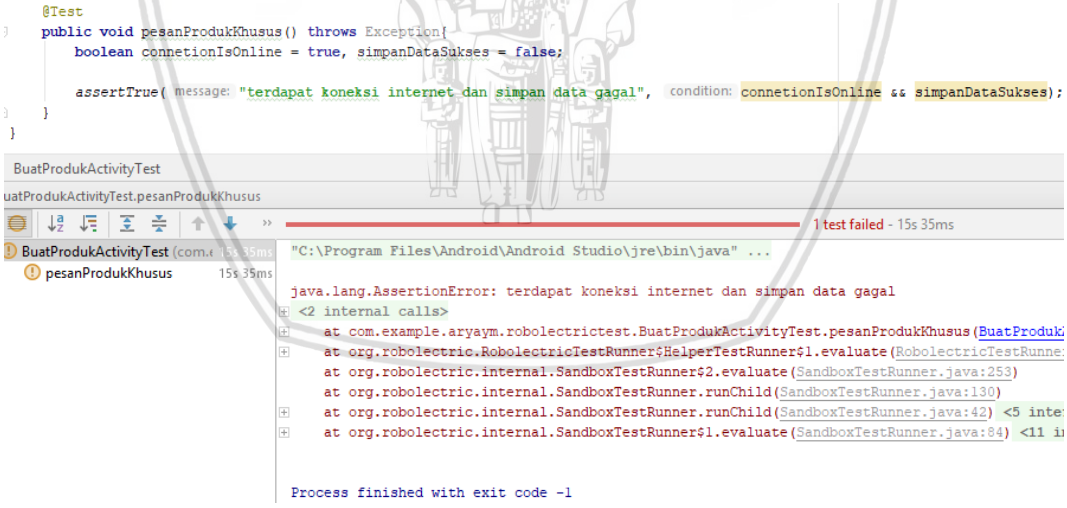
```
1  @Test(expected = Exception.class)
2  public void pesanProdukKhusus() throws Exception{
3      boolean connetionIsOnline = true, simpanDataSukses =
4  true;
5
6      assertTrue("terdapat koneksi internet dan terdapat
7  kegagalan sistem", connetionIsOnline && simpanDataSukses);
8  }
```

Tabel 6.12 Implementasi pengujian pesanProdukKhusus kondisi tidak terdapat koneksi internet

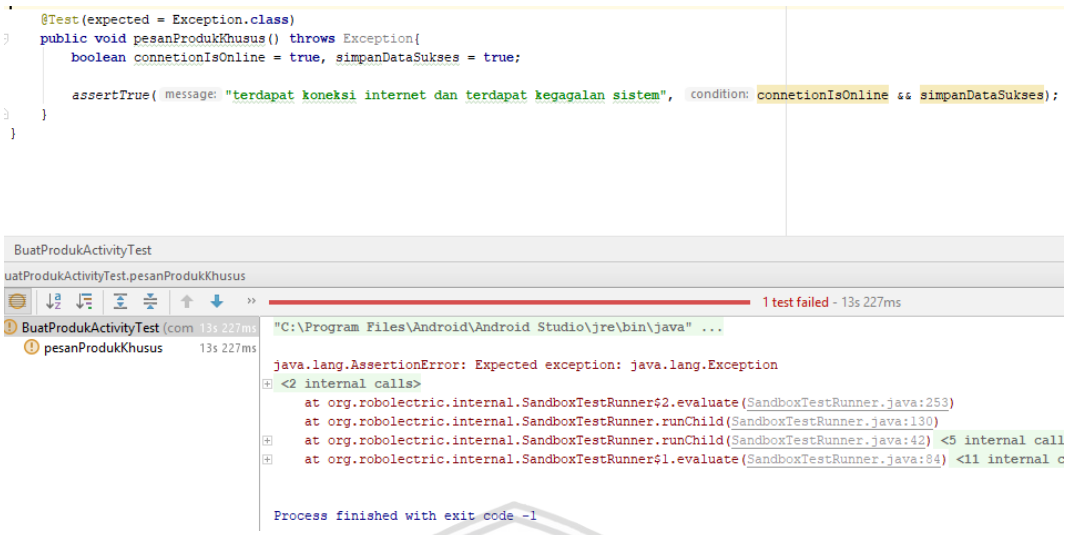
```
1  @Test
2  public void pesanProdukKhusus() throws Exception{
3      boolean connetionIsOnline = false, simpanDataSukses =
4  false;
5
6      assertTrue("tidak terdapat koneksi internet",
7  connetionIsOnline && simpanDataSukses);
8  }
```

c. Hasil pengujian

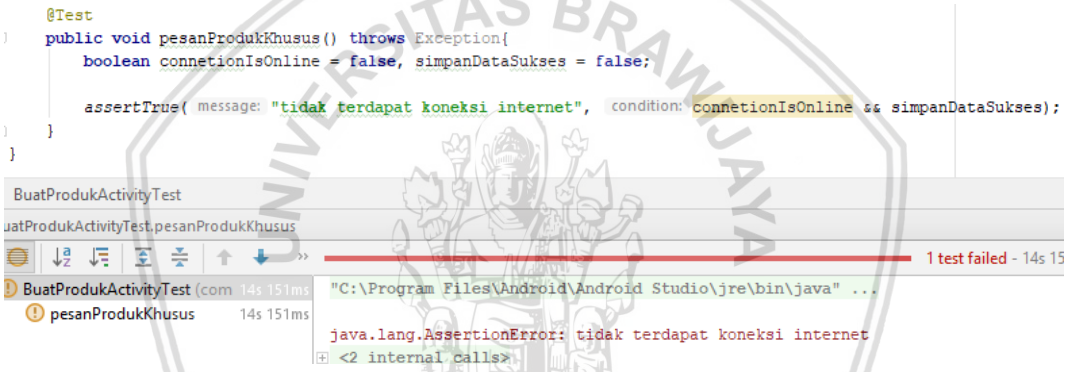
Gambar 6.3 Hasil pengujian pesanProdukKhusus kondisi benar



Gambar 6.4 Hasil pengujian pesanProdukKhusus kondisi simpan data salah



Gambar 6.5 Hasil pengujian pesanProdukKhusus kondisi terdapat kesalahan sistem



Gambar 6.6 Hasil pengujian pesanProdukKhusus kondisi tidak terdapat internet

6.1.1.3 Pengujian Algoritme buatProdukBaru

Pengujian algoritme buatProdukBaru dilakukan untuk menguji alur logic method buatProdukBaru sebelum di implementasikan kedalam kode program. Pada pengujian ini terdapat tiga skenario, dimana skenario pertama mengecek apakah semua kondisi benar, skenario kedua mengecek apabila update gambar gagal dan skenario ketiga mengecek apabila simpan produk gagal.

a. Skenario pengujian

Tabel 6.13 Skenario pengujian buatProdukBaru kondisi semua benar

Nama Kasus Uji	Buat produk baru kondisi semua benar
Tujuan Pengujian	Untuk membuktikan bahwa jika kondisi simpan produk bernilai benar dan update gambar bernilai benar
Prosedur Pengujian	1. inisialisasi nilai variabel

	2. melakukan pembuktian terhadap variabel apakah bernilai benar
Hasil yang Diharapkan	Sistem menampilkan pengujian bernilai benar

Tabel 6.14 Skenario pengujian buatProdukBaru kondisi update gambar gagal

Nama Kasus Uji	Buat produk baru kondisi update gambar gagal
Tujuan Pengujian	Untuk membuktikan bahwa jika kondisi simpan produk bernilai benar dan update gambar bernilai salah
Prosedur Pengujian	1. inisialisasi nilai variabel 2. melakukan pembuktian terhadap variabel apakah bernilai benar
Hasil yang Diharapkan	Sistem menampilkan pengujian bernilai salah

Tabel 6.15 Skenario pengujian buatProdukBaru kondisi simpan produk gagal

Nama Kasus Uji	Buat produk baru kondisi simpan produk gagal
Tujuan Pengujian	Untuk membuktikan bahwa jika kondisi simpan produk bernilai salah dan update gambar bernilai benar
Prosedur Pengujian	1. inisialisasi nilai variabel 2. melakukan pembuktian terhadap variabel apakah bernilai benar
Hasil yang Diharapkan	Sistem menampilkan pengujian bernilai

b. Implementasi skenario pengujian

Tabel 6.16 Implementasi pengujian buatProdukBaru kondisi benar

```
1  @Test
2  public void testBuatProdukBaru() {
3      boolean simpanPordukSukses = true, updateGambarSukses =
4  true;
5
6      assertTrue("simpan produk sukses dan update gambar
7  sukses",simpanPordukSukses && updateGambarSukses);
8  }
```

Tabel 6.17 Implementasi pengujian buatProdukBaru kondisi update gambar gagal

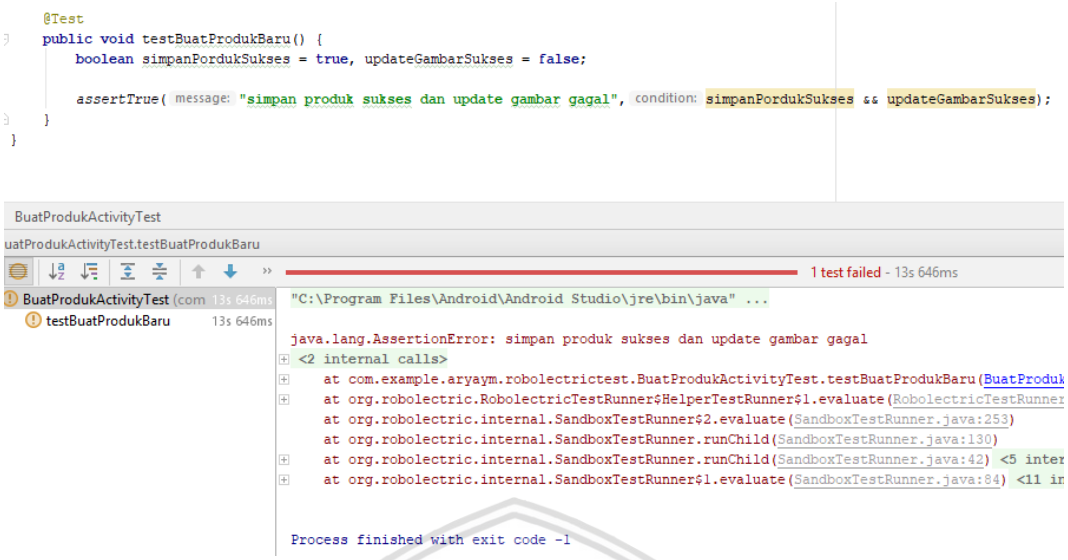
```
1  @Test
2  public void testBuatProdukBaru() {
3      boolean simpanPordukSukses = true, updateGambarSukses =
4  false;
5
6      assertTrue("simpan produk sukses dan update gambar
7  gagal",simpanPordukSukses && updateGambarSukses);
8  }
```

Tabel 6.18 Implementasi pengujian buatProdukBaru kondisi simpan produk gagal

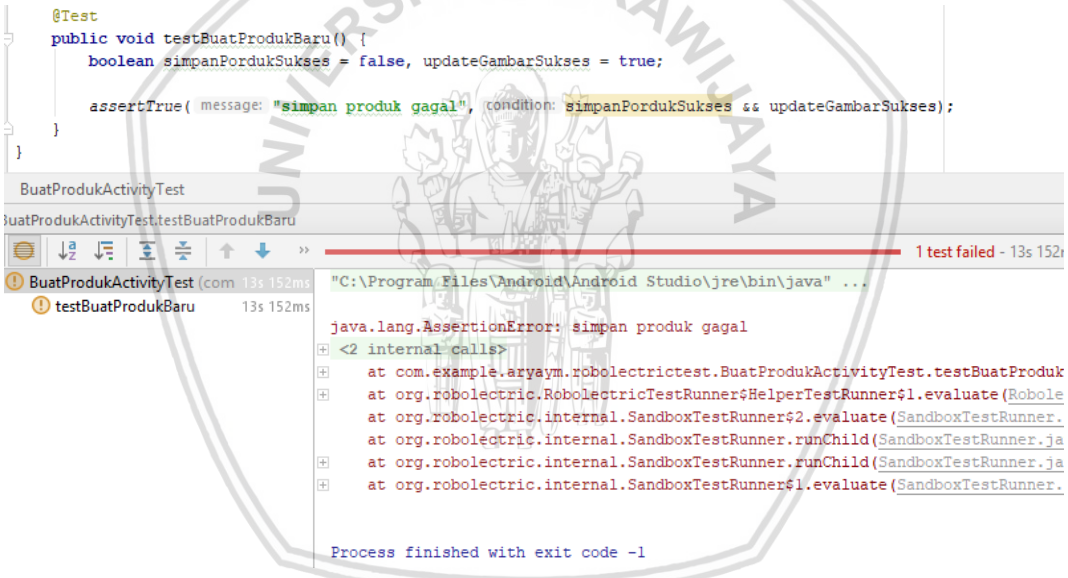
```
1  @Test
2  public void testBuatProdukBaru() {
3      boolean simpanPordukSukses = false, updateGambarSukses =
4  true;
5
6      assertTrue("simpan produk gagal",simpanPordukSukses &&
7  updateGambarSukses);
8  }
```

c. Hasil pengujian

Gambar 6.7 Hasil pengujian buatProdukBaru kondisi



Gambar 6.8 Hasil pengujian buatProdukBaru kondisi



Gambar 6.9 Hasil pengujian buatProdukBaru kondisi

6.1.2 Pengujian Unit

Perangkat lunak yang dikembangkan dengan paradigma *object-oriented programming* menerapkan pengujian unit untuk suatu *method* (operasi) dari suatu *class*. Pada pengujian unit perangkat lunak Mlijo digunakan teknik *White-Box Testing* dengan teknik *Basis Path Testing*. Pada teknik *Basis Path Testing*, proses pengujian dilakukan dengan memodelkan algoritma pada suatu *flow graph*, menentukan jumlah kompleksitas siklomatis (*cyclomatic complexity*), menentukan sebuah basis set dari jalur independen dan memberikan kasus uji (*test case*) pada setiap basis set yang telah ditentukan. Pengujian unit yang dilakukan merupakan lanjutan dari pengujian yang sebelumnya dilakukan, yaitu *test driven development.* Pada pengujian unit ini, dilakukan pada kode program.

6.1.2.1 Pengujian *Method* `cekKolomIsian()`

Method `cekKolomIsian` merupakan unit *method* dari *class* `buatProdukActivity` dan `pesanProdukKhususActivity` yang berfungsi untuk melakukan pengecekkan jika terdapat kolom isian yang masih kosong. Walaupun di implementasikan pada *class* yang berbeda, namun method ini memiliki alur yang sama, oleh karena itu hanya dilakukan pengujian satu kali saja. *Method* `cekKolomIsian` di tunjukan oleh gambar 6.1 berikut :

```
1  private boolean cekKolomIsian() {
       boolean hasil;
       if (TextUtils.isEmpty(inputNamaProduk.getText())
   || TextUtils.isEmpty(inputHargaProduk.getText()) ||
   TextUtils.isEmpty(inputNominalSatuan.getText()) ||
   TextUtils.isEmpty(inputDeskripsiProduk.getText())) {
3          hasil = false;
       } else {
5          hasil = true;
       }
       return hasil;
7  }
```

(2) (4) (6)

Gambar 6.10 Kode Program *Method* cekKolomIsian

Berdasarkan proses pengujian tersebut, Gambar 6.2 menunjukkan *flow graph* dari *method* `cekKolomIsian` yang berfungsi menggambarkan alur dari *method* tersebut.

Gambar 6.11 *Flow Graph Method* cekKolomIsian

Tahapan yang akan dilakukan selanjutnya adalah menghitung *cyclomatic complexity*. Untuk menghitung *cyclomatic complexity* akan digunakan persamaan V(G) = E – N +2 dengan V(G) adalah jumlah kompleksitas siklomatis, lalu E (*edge*) adalah garis penghubung antar node dan N merupakan jumlah simpul (node).

V(G) = 7 – 7 + 2

= 0 + 2

= 2

Berdasarkan nilai *cyclomatic complexity* didapatkan dua buah basis set jalur *independent*, yaitu :

Jalur 1 : 1 – 2 – 3 – 6 – 7

Jalur 2 : 1 – 4 – 5 – 6 – 7

Berdasarkan jumlah jalur yang didapatkam, maka akan dilakukan pengujian terhadap tiap jalur tersebut yang ditunjukkan oleh Tabel 6.1 di bawah ini.

Tabel 6.19 Kasus Uji Pengujian Unit *Method* cekKolomIsian

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Ketika nilai didalam kolom isian nama produk atau harga produk atau nominal satuan atau deksripsi produk kosong.	Menginisialisasi variabel `boolean` hasil menjadi bernilai *false* dan memberikan nilai kembalian *false*.	Menginisialisasi variabel `boolean` hasil menjadi bernilai *false* dan memberikan nilai kembalian *false*.
2	Ketika nilai didalam kolom isian nama produk atau harga produk atau nominal satuan atau deksripsi produk sudah terisi dengan lengkap dan benar.	Menginisialisasi variabel `boolean` hasil menjadi bernilai *true* dan memberikan nilai kembalian *true*.	Menginisialisasi variabel `boolean` hasil menjadi bernilai *true* dan memberikan nilai kembalian *true*.

6.1.2.2 Pengujian *Method* `pesanProdukKhusus()`

Method `pesanProdukKhusus` merupakan unit *method* dari *class* `pesanProdukKhususActivity` yang berfungsi untuk melakukan proses pemesanan produk yang tidak terdapat dalam daftar produk. Method ini di panggil oleh method `buatProdukKhusus` untuk melakukan simpan data pemesanan kedalam database. *Method* `pesanProdukKhusus` ditunjukan oleh gambar 6.3 berikut.

```
private void pesanProdukKhusus(String produkId) {
    String idProduk = produkId;
    String pesanKonsumen =
notePenjual.getText().toString();
    String tanggalKirim=
inputTanggalKirim.getText().toString();
    String waktuKirim =
inputJamKirim.getText().toString();
    long orderTime = new Date().getTime();
    if
(InternetConnection.getInstance().isOnline(PesanProdu
kKhususActivity.this)) {
        try {
```

1

2

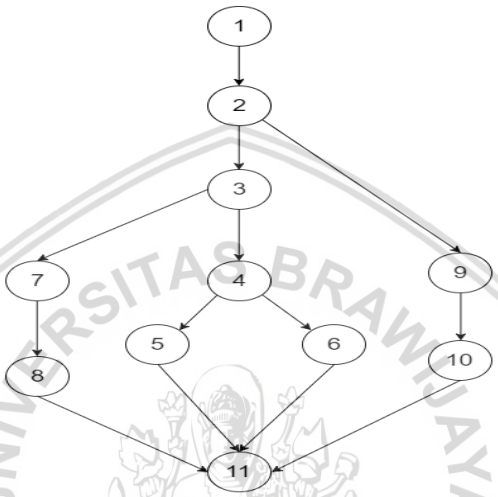
3


String pushId =
mDatabase.child(Constants.KONSUMEN).child(getUid()).child(Constants.DAFTAR_TRANSAKSI).push().getKey();
String transaksiId = pushId;
Map<String, Object> pemesanan = new HashMap<>();
pemesanan.put(Constants.BIAYA_KIRIM, 0);
pemesanan.put(Constants.NOTE_KONSUMEN,pesanKonsumen);
pemesanan.put(Constants.ID_KONSUMEN, getUid());
pemesanan.put(Constants.ID_TRANSAKSI, transaksiId);
pemesanan.put(Constants.ID_PENJUAL, penjualId);
pemesanan.put(Constants.ID_PRODUK, idProduk);
pemesanan.put(Constants.JUMLAH_ORDER_PRODUK, 1);
pemesanan.put(Constants.NAMA_PENERIMA, " ");
pemesanan.put(Constants.STATUS_TRANSAKSI, 1);
pemesanan.put(Constants.JENIS_PRODUK, Constants.PRODUK_KHUSUS);
pemesanan.put(Constants.JUMLAH_HARGA_PRODUK, 0);
pemesanan.put(Constants.TANGGAL, orderTime);
pemesanan.put(Constants.TANGGAL_KIRIM, tanggalKirim);
pemesanan.put(Constants.WAKTU_KIRIM, waktuKirim);
mDatabase.child(Constants.KONSUMEN).child(getUid()).child(Constants.DAFTAR_TRANSAKSI).child(Constants.PEMBELIAN_BARU).child(transaksiId).setValue(pemesanan);
mDatabase.child(Constants.PENJUAL).child(penjualId).child(Constants.DAFTAR_TRANSAKSI).child(Constants.PENJUALAN_BARU).child(transaksiId).setValue(pemesanan)
.addOnSuccessListener(new OnSuccessListener<Void>() {
@Override
public void onSuccess(Void aVoid) {
finish();
Toast.makeText(getApplicationContext(), "Anda telah berhasil melakukan pemesanan produk", Toast.LENGTH_SHORT).show();
}
}).addOnFailureListener(new OnFailureListener() {
@Override
public void onFailure(@NonNull Exception e) {
ShowSnackbar.showSnack(PesanProdukKhususActivity.this, "gagal membuat pesanan, silahkan ulangi lagi");
}
});
} catch (Exception e) {
ShowSnackbar.showSnack(this, getResources().getString(R.string.msg_error));
}
} else {
final Snackbar snackbar = Snackbar.make(activity, getResources().getString(R.string.msg_noInternet), Snackbar.LENGTH_INDEFINITE);
snackbar.setAction(getResources().getString(R.string.ok), new View.OnClickListener() {
@Override
public void onClick(View v) {
4
5
6
7
8
9
10

```
                    snackbar.dismiss();
                }
            });
            snackbar.show();
        }
}
```
11

Gambar 6.12 Kode Program *Method* `pesanProdukKhusus`

Berdasarkan proses pengujian tersebut, Gambar 6.4 menunjukkan *flow graph* dari *method* `pesanProdukKhusus` yang berfungsi menggambarkan alur dari *method* tersebut.



Gambar 6.13 *Flow Graph Method* `pesanProdukKhusus`

Tahapan yang akan dilakukan selanjutnya adalah menghitung *cyclomatic complexity*. Untuk menghitung *cyclomatic complexity* akan digunakan persamaan V(G) = E – N +2 dengan V(G) adalah jumlah kompleksitas siklomatis, lalu E (*edge*) adalah garis penghubung antar node dan N merupakan jumlah simpul (node).

V(G) = 13 – 11 + 2

= 2 + 2

= 4

Berdasarkan nilai *cyclomatic complexity* didapatkan tiga buah basis set jalur *independent*, yaitu :

Jalur 1 : 1 – 2 – 3 – 4 – 5 – 11

Jalur 2 : 1 – 2 – 3 – 4 – 6 – 11

Jalur 3 : 1 – 2 – 3 – 7 – 8 – 11

Jalur 4 : 1 – 2 – 9 – 10 – 11

Berdasarkan jumlah jalur yang didapatkam, maka akan dilakukan pengujian terhadap tiap jalur tersebut yang ditunjukkan oleh Tabel 6.2 di bawah ini.

Tabel 6.20 Kasus Uji Pengujian Unit *Method* pesanProdukKhusus

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Ketika terdapat koneksi internet, lalu tidak ada kegagalan dari sistem dan proses simpan data sukses.	Melakukan proses buat produk khusus dan menyimpannya didalam *firebase database* serta menampilkan pesan pemesanan berhasil.	Melakukan proses buat produk khusus dan menyimpannya didalam *firebase database* serta menampilkan pesan pemesanan berhasil.
2	Ketika terdapat koneksi internet, lalu tidak ada kegagalan dari sistem dan proses simpan data gagal.	Melakukan proses buat produk khusus dan gagal menyimpannya didalam *firebase database* serta menampilkan pesan pemesanan gagal.	Melakukan proses buat produk khusus dan gagal menyimpannya didalam *firebase database* serta menampilkan pesan pemesanan gagal.
3	Ketika terdapat koneksi internet, lalu terdapat kegagalan dari sistem.	Menampilkan pesan *error*.	Menampilkan pesan *error*.
4	Ketika tidak ada koneksi internet.	Menampilkan pesan bahwa tidak terkoneksi internet.	Menampilkan pesan bahwa tidak terkoneksi internet.

6.1.2.3 Pengujian *Method* `buatProdukBaru()`

Method `buatProdukBaru` merupakan unit *method* dari *class* `buatProdukActivity` yang berfungsi untuk melakukan proses penyimpanan data produk ke dalam database. Method ini di panggil oleh method `simpanProduk`. *Method* `buatProdukBaru` ditunjukan oleh gambar 6.5 berikut.

```
public void buatProdukBaru(final ProdukModel
produkModel, ArrayList<Uri> imageUri){
    final String pushId =
mFirestore.collection(Constants.PRODUK_REGULER).docum
ent().getId();
    String produkId = pushId;
HashMap<String, Object> dataProduk = new HashMap<>();
    dataProduk.put(Constants.ID_PENJUAL,
produkModel.getUid());
    dataProduk.put(Constants.WAKTU_DIBUAT,
produkModel.getWaktuDibuat());
    dataProduk.put(Constants.NAMAPRODUK,
produkModel.getNamaProduk());
    dataProduk.put(Constants.ID_KATEGORI,
produkModel.getKategoriProduk());
    dataProduk.put(Constants.HARGAPRODUK,
```

1

```
produkModel.getHargaProduk());
    dataProduk.put(Constants.DIGITSATUAN,
produkModel.getSatuanProduk());
    dataProduk.put(Constants.NAMASATUAN,
produkModel.getNamaSatuan());
    dataProduk.put(Constants.ID_PRODUK, produkId);
    dataProduk.put(Constants.GAMBARPRODUK,
produkModel.getGambarProduk());
    dataProduk.put(Constants.ID_LOKASI,
produkModel.getIdLokasi());
    dataProduk.put(Constants.DESKRIPSI,
produkModel.getDeskripsiProduk());

mFirestore.collection(Constants.PRODUK_REGULER).docum
ent(produkId).set(dataProduk).addOnSuccessListener(ne
w OnSuccessListener<Void>() {
    @Override
    public void onSuccess(Void aVoid) {
        UploadPhotoThreadListener
uploadPhotoThreadListener = new
UploadPhotoThreadListener() {
            @Override
            public void
onUploadPhotoSuccess(ArrayList<String> photoUrls) {
  Map<String, Object> gambarProduk = new HashMap<>();
gambarProduk.put(Constants.GAMBARPRODUK, photoUrls);

mFirestore.collection(Constants.PRODUK_REGULER).docum
ent(pushId).update(gambarProduk)
        .addOnSuccessListener(new
OnSuccessListener<Void>() {
    @Override
    public void onSuccess(Void aVoid) {
        view.hideProgressDialog();
        view.finish();
        Toast.makeText(view.getApplicationContext(),
"Produk anda telah tersimpan",
Toast.LENGTH_SHORT).show();
    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        view.hideProgressDialog();
        view.finish();
        Toast.makeText(view.getApplicationContext(),
        "Maaf, gagal melakukan upload gambar produk.
Silahkan ulangi kembali!",Toast.LENGTH_SHORT).show();
    }
});
            }
        };
        new UploadPhotoThread(pushId, imageUri,
uploadPhotoThreadListener).execute();
    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        Toast.makeText(view.getApplicationContext(),
                "Maaf, gagal menyimpan produk.
```

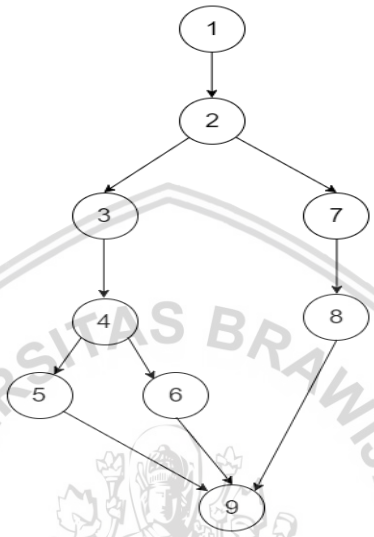
1 2 3 4 5 6 7 8

```
        Silahkan ulangi kembali", Toast.LENGTH_SHORT).show();
            }
        });
    }
```

9

Gambar 6.14 Kode Program `buatProdukBaru`

Berdasarkan proses pengujian tersebut, Gambar 6.6 menunjukkan *flow graph* dari *method* `buatProdukBaru` yang berfungsi menggambarkan alur dari *method* tersebut.



Gambar 6.15 *Flow Graph Method* `buatProdukBaru`

Tahapan yang akan dilakukan selanjutnya adalah menghitung *cyclomatic complexity*. Untuk menghitung *cyclomatic complexity* akan digunakan persamaan V(G) = E – N +2 dengan V(G) adalah jumlah kompleksitas siklomatis, lalu E (*edge*) adalah garis penghubung antar node dan N merupakan jumlah simpul (node).

V(G) = 10 – 9 + 2

= 1 + 2

= 3

Berdasarkan nilai *cyclomatic complexity* didapatkan tiga buah basis set jalur *independent*, yaitu :

Jalur 1 : 1 – 2 – 3 – 4 – 5

Jalur 2 : 1 – 2 – 3 – 4 – 6

Jalur 3 : 1 – 2 – 7 – 8 – 9

Berdasarkan jumlah jalur yang didapatkam, maka akan dilakukan pengujian terhadap tiap jalur tersebut yang ditunjukkan oleh Tabel 6.3 di bawah ini.

Tabel 6.21 Kasus Uji Pengujian Unit *Method* buatProdukBaru

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Ketika berhasil melakukan simpan	Menampilkan pesan dialog bahwa berhasil	Menampilkan pesan dialog bahwa berhasil

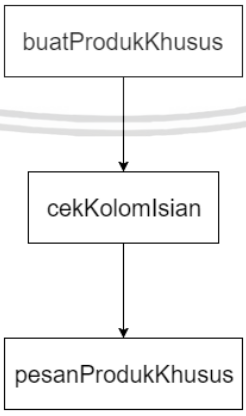
	data kedalam *database* dan juga berhasil melakukan *update* data gambar produk.	melakukan simpan produk.	melakukan simpan produk.
2	Ketika berhasil melakukan simpan data kedalam *database* dan gagal melakukan *update* data gambar produk.	Menampilkan pesan dialog bahwa gagal melakukan *upload* gambar produk.	Menampilkan pesan dialog bahwa gagal melakukan *upload* gambar produk.
3	Ketika gagal melakukan simpan data kedalam *database*.	Menampilkan pesan dialog bahwa gagal melakukan simpan produk.	Menampilkan pesan dialog bahwa gagal melakukan simpan produk.

6.1.3 Pengujian Integrasi

Pengujian integrasi diterapkan pada proses yang mengintegrasikan fungsionalitas dari beberapa *class* atau *method* untuk melakukan sebuah operasi tertentu. Pada pengujian integrasi ini digunakan pendeketan *top down*.

6.1.3.1 Pengujian *Method* `buatProdukKhusus()`

Method `buatProdukKhusus` merupakan suatu *method* integrasi yang berfungsi untuk membuat produk khusus sebagai syarat alur pemesanan produk khusus. *Method* ini terintegrasi dengan *method* `cekKolomIsian` dan `pesanProdukKhusus` yang telah di uji pada pengujian unit. Diagram hiraki dari *method* yang saling terintegrasi tersebut di gambarkan oleh gambar 6.7 berikut :



Gambar 6.16 Diagram hiraki Pengujian Integrasi *Method* `buatProdukKhusus`

Strategi pengujian integrasi yang akan digunakan untuk menguji *method* `cekKolomIsisan` adalah menggunakan pendekatan *Top Down*. Pada pendekatan ini pada *method* `buatProdukKhusus` akan di buat sebuah *stub*

yang berfungsi sebagai modul pengganti yang akan memanggil modul yang sedang diuji. Sedangkan untuk menguji *method* `pesanProdukKhusus`, hasil dari IS1 akan di buat sebuah *stub* yang berfungsi sebagai modul pengganti yang akan memanggil modul yang sedang di uji. Langkah – langkah dalam pengujian integrasi *method* `buatProdukKhusus` seperti berikut ini :

Langkah 1 : IS1 = `buatProdukKhusus + cekKolomIsian`

Langkah 2 : IS2 = IS1 + `pesanProdukKhusus`

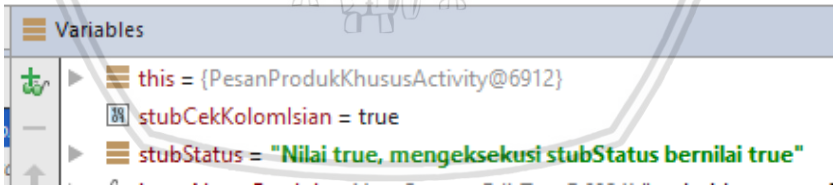
Pada langkah pertama pengujian *method* integrasi `buatProdukKhusus` dan `cekKolomIsian` akan menggunakan *stub*. *Stub* yang akan digunakan pada pengujian *method* `cekKolomIsian` diberikan nilai *true* atau *false* yang merupakan tipe data `boolean`. Gambar 6.8 merupakan kode program *method* `buatProdukKhusus` yang menggunakan stub dengan nilai *true* sebagai modul pengganti nilai kembalian *method* cekKolomIsian.

```
private void buatProdukKhusus() {
        boolean stubCekKolomIsian = true;
        if (stubCekKolomIsian) {
            String stubStatus = "Nilai true,
mengeksekusi stubStatus bernilai true";
            String namaProduk =
inputNamaProduk.getText().toString();
            int satuanProduk =
Integer.parseInt(inputSatuanDigit.getText().toString(
));
if(InternetConnection.getInstance().isOnline(PesanPro
dukKhususActivity.this)) {
                try {
                    String pushId =
mDatabase.child(Constants.PRODUK_KHUSUS).child(kategor
iProduk).push().getKey();
        final String produkId = pushId;
HashMap<String, Object> dataProduk = new HashMap<>();
dataProduk.put(Constants.NAMAPRODUK, namaProduk);
dataProduk.put(Constants.DIGITSATUAN, satuanProduk);
dataProduk.put(Constants.NAMASATUAN, namaSatuan);
dataProduk.put(Constants.ID_PRODUK, produkId);

mFirestore.collection(Constants.PRODUK_KHUSUS).docume
nt(produkId).set(dataProduk)
                            .addOnSuccessListener(new
OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {
pesanProdukKhusus(produkId);
                        }
                    }).addOnFailureListener(new
OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull
Exception e) {
ShowSnackbar.showSnack(PesanProdukKhususActivity.this
, "gagal membuat produk pesanan, silahkan ulangi
lagi");
```

```
                        }
                    });
                } catch (Exception e) {
                    ShowSnackbar.showSnack(this,
getResources().getString(R.string.msg_error));
                }
            } else {
                final Snackbar snackbar =
Snackbar.make(activity,
getResources().getString(R.string.msg_noInternet),
Snackbar.LENGTH_INDEFINITE);
snackbar.setAction(getResources().getString(R.string.
ok), new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        snackbar.dismiss();
                    }
                });
                snackbar.show();
            }
        } else {
            String stubStatus = "Nilai false,
mengeksekusi stubStatus bernilai false";
            ShowAlertDialog.showAlert("Anda harus
mengisi semua Form yang tersedia !", this);
        }
    }
```

Gambar 6.17 Kode Program Stub *Method* `stubCekKolomIsian` *True*

Hasil dari penggunaan stub dengan nilai variabel `stubCekKolomIsian` *true* adalah *method* `buatProdukKhusus` berhasil mengeksekusi nilai `String stubStatus` yang merupakan *statement* pada kondisi jika `stubCekKolomIsian` bernilai *true*. Hal ini ditunjukkan oleh Gambar 6.9 dibawah ini.



Gambar 6.18 Hasil Pengujian Menggunakan Stub *Method* `stubCekKolomIsian` *True*

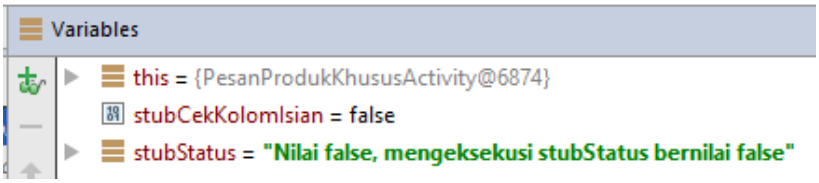
Selanjutnya adalah mengganti nilai `stubCekKolomIsian` dengan nilai *false* sebagai modul pengganti nilai kembalian *method* `cekKolomIsian`. Gambar 6.10 menunjukkan penggunaan *stub* dengan nilai *stub* sama dengan *false*.

```
private void buatProdukKhusus() {
        boolean stubCekKolomIsian = false;
        if (stubCekKolomIsian) {
            String stubStatus = "Nilai true,
mengeksekusi stubStatus bernilai true";
            String namaProduk =
inputNamaProduk.getText().toString();
            int satuanProduk =
```

```
Integer.parseInt(inputSatuanDigit.getText().toString(
));
if(InternetConnection.getInstance().isOnline(PesanPro
dukKhususActivity.this)) {
                try {
                    String pushId =
mDatabase.child(Constants.PRODUK_KHUSUS).child(kategor
iProduk).push().getKey();
final String produkId = pushId;
HashMap<String, Object> dataProduk = new HashMap<>();
dataProduk.put(Constants.NAMAPRODUK, namaProduk);
dataProduk.put(Constants.DIGITSATUAN, satuanProduk);
dataProduk.put(Constants.NAMASATUAN, namaSatuan);
dataProduk.put(Constants.ID_PRODUK, produkId);

mFirestore.collection(Constants.PRODUK_KHUSUS).docume
nt(produkId).set(dataProduk)
                                .addOnSuccessListener(new
OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {
pesanProdukKhusus(produkId);
                        }
                    }).addOnFailureListener(new
OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull
Exception e) {
ShowSnackbar.showSnack(PesanProdukKhususActivity.this
, "gagal membuat produk pesanan, silahkan ulangi
lagi");
                        }
                    });
                } catch (Exception e) {
                    ShowSnackbar.showSnack(this,
getResources().getString(R.string.msg_error));
                }
            } else {
                final Snackbar snackbar =
Snackbar.make(activity,
getResources().getString(R.string.msg_noInternet),
Snackbar.LENGTH_INDEFINITE);
snackbar.setAction(getResources().getString(R.string.
ok), new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        snackbar.dismiss();
                    }
                });
                snackbar.show();
            }
        } else {
            String stubStatus = "Nilai false,
mengeksekusi stubStatus bernilai false";
            ShowAlertDialog.showAlert("Anda harus
mengisi semua Form yang tersedia !", this);
        }
    }
```

Gambar 6.19 Kode Program Stub *Method* `stubCekKolomIsian` *False*

Hasil dari penggunaan *stub* dengan nilai variabel `stubCekKolomIsian` *false* adalah *method* `buatProdukKhusus` berhasil mengeksekusi nilai `String stubStatus` yang merupakan statement pada kondisi jika `stubCekKolomIsian` bernilai *false*. Hal ini ditunjukkan oleh Gambar 6.11.



Gambar 6.20 Hasil Pengujian Menggunakan Stub *Method* `stubCekKolomIsian` *False*

Kemudian dilanjutkan dengan pengujian integrasi langkah kedua. Dimana pada pengujian integrasi kedua *method* `buatProdukKhusus` dan `cekKolomIsian` terintegrasi dengan *method* `pesanProdukKhusus`. Pada pengujian ini menggunakan pendekatan *Top Down* yang mana menggunakan *stub*. *Stub* yang digunakan pada pengujian *method* `buatProdukKhusus` berupa *method dummy* `stubPesanProdukKhusus` yang diberikan nilai *dummy* dan fungsi seperti pada *method* `pesanProdukKhusus`. Gambar 6.12 merupakan kode program *method* `buatProdukKhusus` yang menggunakan stub *method* `stubPesanProdukKhusus`.

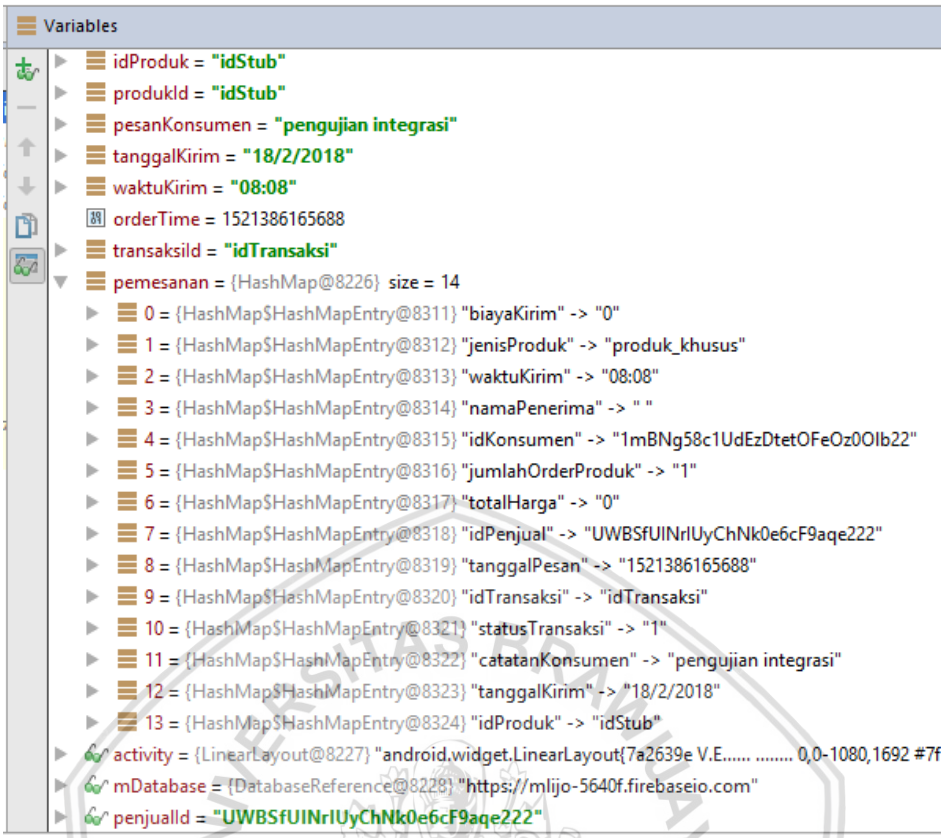
```
private void buatProdukKhusus() {
        if (cekKolomIsian()) {
            String namaProduk = "produk stub";
            int satuanProduk = 1;
            if
(InternetConnection.getInstance().isOnline(PesanProdu
kKhususActivity.this)) {
                try {
                    final String produkId = "idStub";
                    HashMap<String, Object>
dataProduk = new HashMap<>();

dataProduk.put(Constants.NAMAPRODUK, namaProduk);
dataProduk.put(Constants.DIGITSATUAN, satuanProduk);
dataProduk.put(Constants.NAMASATUAN, "Gram(Gr)");
dataProduk.put(Constants.ID_PRODUK, produkId);

mFirestore.collection(Constants.PRODUK_KHUSUS).docume
nt(produkId).set(dataProduk)
                                .addOnSuccessListener(new
OnSuccessListener<Void>() {
                            @Override
                            public void onSuccess(Void
aVoid) {
stubPesanProdukKhusus(produkId);
                            }
                        }).addOnFailureListener(new
OnFailureListener() {
                            @Override
```

```
                            public void
onFailure(@NonNull Exception e) {
ShowSnackbar.showSnack(PesanProdukKhususActivity.this
, "gagal membuat produk pesanan, silahkan ulangi
lagi");
                            }
                        });
                    } catch (Exception e) {
                        ShowSnackbar.showSnack(this,
getResources().getString(R.string.msg_error));
                    }
                } else {
                    final Snackbar snackbar =
Snackbar.make(activity,
getResources().getString(R.string.msg_noInternet),
Snackbar.LENGTH_INDEFINITE);

snackbar.setAction(getResources().getString(R.string.
ok), new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                            snackbar.dismiss();
                        }
                    });
                    snackbar.show();
                }
            } else {
                ShowAlertDialog.showAlert("Anda harus
mengisi semua Form yang tersedia !", this);
            }
        }
//    stubPesanProdukKhusus
private void stubPesanProdukKhusus(String idProduk) {
            String produkId = idProduk;
            String pesanKonsumen = "pengujian integrasi";
            String tanggalKirim = "18/2/2018";
            String waktuKirim = "08:08";
            long orderTime = new Date().getTime();

            if
(InternetConnection.getInstance().isOnline(PesanProdu
kKhususActivity.this)) {
                try {
         String transaksiId = "idTransaksi";
         Map<String, Object> pemesanan = new HashMap<>();
pemesanan.put(Constants.BIAYA_KIRIM, 0);
pemesanan.put(Constants.NOTE_KONSUMEN,
pesanKonsumen);
pemesanan.put(Constants.ID_KONSUMEN, getUid());
pemesanan.put(Constants.ID_TRANSAKSI, transaksiId);
pemesanan.put(Constants.ID_PENJUAL, penjualId);
pemesanan.put(Constants.ID_PRODUK, idProduk);
pemesanan.put(Constants.JUMLAH_ORDER_PRODUK, 1);
pemesanan.put(Constants.NAMA_PENERIMA, " ");
pemesanan.put(Constants.STATUS_TRANSAKSI, 1);
pemesanan.put(Constants.JENIS_PRODUK,
Constants.PRODUK_KHUSUS);
pemesanan.put(Constants.JUMLAH_HARGA_PRODUK, 0);
pemesanan.put(Constants.TANGGAL, orderTime);
```

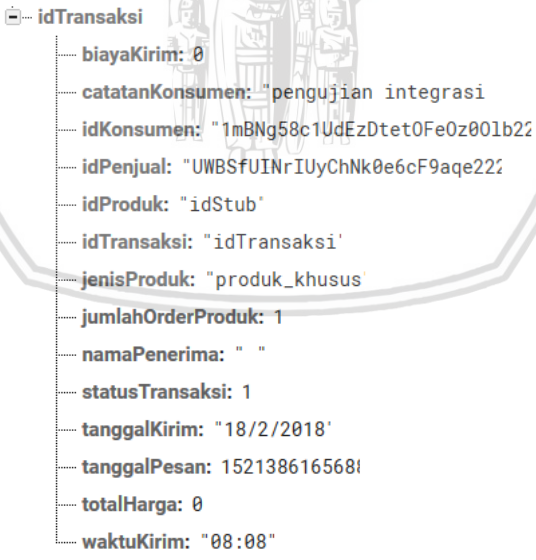
```
pemesanan.put(Constants.TANGGAL_KIRIM, tanggalKirim);
pemesanan.put(Constants.WAKTU_KIRIM, waktuKirim);

mDatabase.child(Constants.KONSUMEN).child(getUid()).c
hild(Constants.DAFTAR_TRANSAKSI).child(Constants.PEMB
ELIAN_BARU).child(transaksiId).setValue(pemesanan);
mDatabase.child(Constants.PENJUAL).child(penjualId).c
hild(Constants.DAFTAR_TRANSAKSI).child(Constants.PENJ
UALAN_BARU).child(transaksiId).setValue(pemesanan)
                        .addOnSuccessListener(new
OnSuccessListener<Void>() {
                            @Override
                            public void
onSuccess(Void aVoid) {
                             buatNotifikasiOrder();
                             finish();
Toast.makeText(getApplicationContext(), "Anda telah
berhasil melakukan pemesanan produk",
Toast.LENGTH_SHORT).show();
                            }
                        }).addOnFailureListener(new
OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull
Exception e) {

ShowSnackbar.showSnack(PesanProdukKhususActivity.this
, "gagal membuat pesanan, silahkan ulangi lagi");
                    }
                });
                String statusPemesanan = "pemesanan
produk berhasil";
            } catch (Exception e) {
                ShowSnackbar.showSnack(this,
getResources().getString(R.string.msg_error));
            }
        } else {
            final Snackbar snackbar =
Snackbar.make(activity,
getResources().getString(R.string.msg_noInternet),
Snackbar.LENGTH_INDEFINITE);

snackbar.setAction(getResources().getString(R.string.
ok), new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    snackbar.dismiss();
                }
            });
            snackbar.show();
        }
    }
```

Gambar 6.21 Kode Program Stub *Method* `stubPesanProdukKhusus`

Hasil dari penggunaan *stub* dengan *method* `stubPesanProdukKhusus` adalah *method* `buatProdukKhusus` berhasil mengeksekusi *method* `stubPesanProdukKhusus` dan dapat menjalankan fungsi melakukan pemesanan. Hal ini ditunjukkan oleh Gambar 6.13 dan 6.14.



Gambar 6.22 Hasil *Submit* `stubPesanProdukKhusus` pada `Console`



Gambar 6.23 Hasil *Submit* `stubPesanProdukKhusus` pada `Firebase`

Selanjutnya adalah dengan mengintegrasikan *method* `buatProdukKhusus` dengan *method* `cekKolomIsian` dan `pesanProdukKhusus`. Gambar 6.15 akan menjelaskan tentang kode program *method* `buatProdukKhusus` secara keseluruan yang telah di integrasikan dengan *method* `cekKolomIsian` dan `pesanProdukKhusus`.

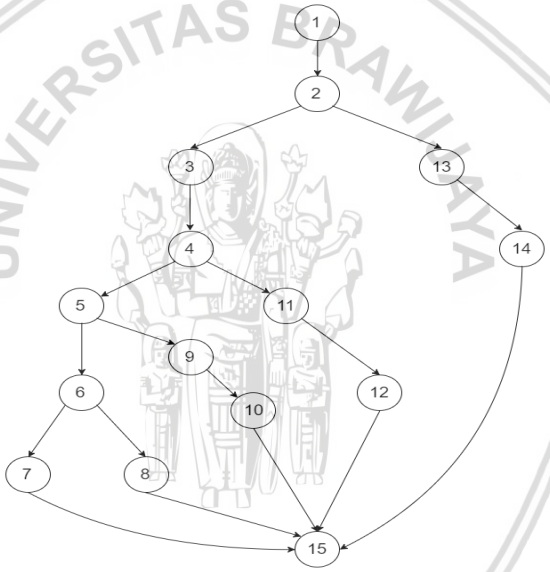
```
private void buatProdukKhusus() {
        if (cekKolomIsian()) {
            String namaProduk =
inputNamaProduk.getText().toString();
            int satuanProduk =
Integer.parseInt(inputSatuanDigit.getText().toString(
));

if(InternetConnection.getInstance().isOnline(PesanPro
dukKhususActivity.this)) {
                try {
                    showProgessDialog();
                    String pushId =
mDatabase.child(Constants.PRODUK_KHUSUS).child(katego
riProduk).push().getKey();
                    final String produkId = pushId;
HashMap<String, Object> dataProduk = new HashMap<>();

dataProduk.put(Constants.NAMAPRODUK, namaProduk);
dataProduk.put(Constants.DIGITSATUAN, satuanProduk);
dataProduk.put(Constants.NAMASATUAN, namaSatuan);
dataProduk.put(Constants.ID_PRODUK, produkId);

mFirestore.collection(Constants.PRODUK_KHUSUS).docume
nt(produkId).set(dataProduk)
                            .addOnSuccessListener(new
OnSuccessListener<Void>() {
                        @Override
                        public void onSuccess(Void aVoid) {

pesanProdukKhusus(produkId);
String statusCekKolomIsian =
String.valueOf(cekKolomIsian());
String status = "cekKolomIsian true, setValue
dataProduk berhasil";
                        }
                    }).addOnFailureListener(new
OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull
Exception e) {
ShowSnackbar.showSnack(PesanProdukKhususActivity.this
, "gagal membuat produk pesanan, silahkan ulangi
lagi");
                        }
                    });
                } catch (Exception e) {
                    ShowSnackbar.showSnack(this,
getResources().getString(R.string.msg_error));
                }
            } else {
                final Snackbar snackbar =
Snackbar.make(activity,
getResources().getString(R.string.msg_noInternet),
Snackbar.LENGTH_INDEFINITE);
snackbar.setAction(getResources().getString(R.string.
ok), new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
```

```
                        snackbar.dismiss();
                    }
                });
                snackbar.show();
            }
        } else {
String statusCekKolomIsian =
String.valueOf(cekKolomIsian());
String status = "cekKolomIsian false, setValue
dataProduk tidak tereksekusi";
            ShowAlertDialog.showAlert("Anda harus
mengisi semua Form yang tersedia !", this);
        }
    }
```

12 14 13 15

Gambar 6.24 Kode Program *Method* `buatProdukKhusus`

Berdasarkan proses pengujian tersebut, Gambar 6.16 menunjukkan *flow graph* dari *method* `buatProdukKhusus` yang berfungsi menggambarkan alur dari *method* tersebut.



Gambar 6.25 *Flow Graph Method* `buatProdukKhusus`

Tahapan yang akan dilakukan selanjutnya adalah menghitung *cyclomatic complexity*. Untuk menghitung *cyclomatic complexity* akan digunakan persamaan V(G) = E – N +2 dengan V(G) adalah jumlah kompleksitas siklomatis, lalu E (*edge*) adalah garis penghubung antar node dan N merupakan jumlah simpul (node).

V(G) = 14 – 12 + 2

= 2+ 2

= 4

Berdasarkan nilai *cyclomatic complexity* didapatkan dua buah basis set jalur *independent*, yaitu :

Jalur 1 : 1 – 2 – 3 – 4 – 5 – 6 – 7 – 15

Jalur 2 : 1 – 2 – 3 – 4 – 5 – 6 – 8 – 15

Jalur 3 : 1 – 2 – 3 – 4 – 5 – 9 – 10 – 15

Jalur 4 : 1 – 2 – 3 – 4 – 11 – 12 – 15

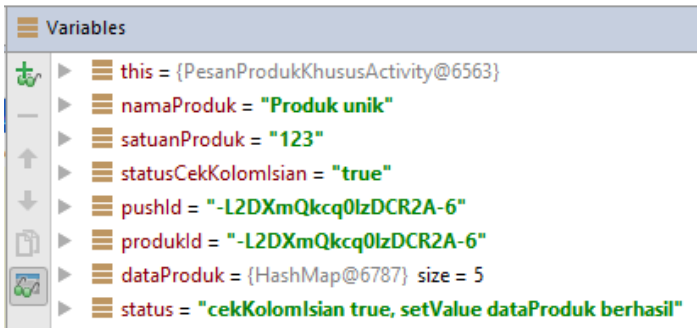
Jalur 5 : 1 – 2 – 13 – 14 – 15

Berdasarkan jumlah jalur yang didapatkan, maka akan dilakukan pengujian terhadap tiap jalur tersebut yang ditunjukkan oleh Tabel 6.4.

Tabel 6.22 Kasus Uji Pengujian Integrasi *Method* `buatProdukKhusus`

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Ketika semua kolom terisisi, terdapat koneksi internet, lalu tidak ada kegagalan dari sistem dan proses simpan data ke database sukses.	Melakukan proses buat produk khusus dan menyimpan ke dalam firebase database dan memanggil *method* `pesanprodukKhusus()`	Melakukan proses buat produk khusus dan menyimpan ke dalam firebase database dan memanggil *method* `pesanprodukKhusus()`
2	Ketika semua kolom terisisi, terdapat koneksi internet, lalu tidak ada kegagalan dari sistem dan proses simpan data ke database gagal.	Menampilkan pesan bahwa proses simpan data ke database error.	Menampilkan pesan bahwa proses simpan data ke database error.
3	Ketika semua kolom terisisi, terdapat koneksi internet, lalu terdapat kegagalan dari sistem	Menampilkan pesan *error*.	Menampilkan pesan *error*.
4	Ketika semua kolom terisi, tidak terdapat koneksi internet.	Menampilkan pesan bahwa tidak terkoneksi internet.	Menampilkan pesan bahwa tidak terkoneksi internet.
5	Ketika nilai *method* cekKolomIsian false.	Menampilkan pesan bahwa form harus di isi semua.	Menampilkan pesan bahwa form harus di isi semua.

Selanjutnya adalah membandingkan nilai yang didapatkan sebelumnya menggunakan *stub* dengan nilai yang didapatkan dengan menggunakan *method* `cekKolomIsian` yang telah diintegrasikan dengan *method* `buatProdukKhusus`. Untuk mendapatkan nilai *true* dari nilai kembalian *method* `cekKolomIsian`, akan dilakukan simulasi dengan mengisi nilai kolom isian dengan suatu nilai agar mendapat nilai *true*. Gambar 6.17 menunjukkan hasil integrasi antara *method* `buatProdukKhusus` dan *method* `cekKolomIsian` dengan nilai kembalian dari *method* `cekKolomIsian` bernilai *true*.

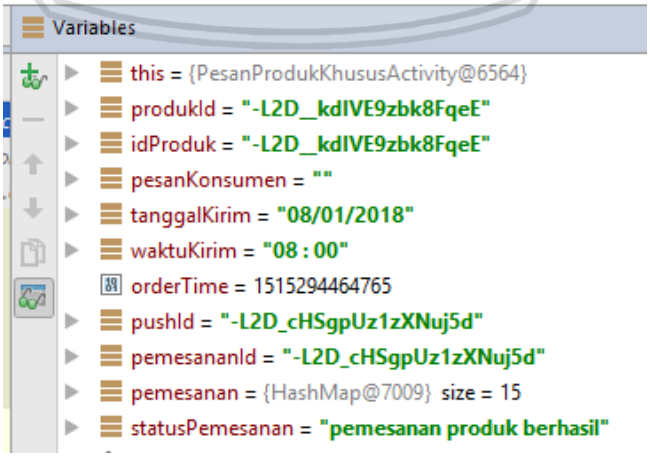


Gambar 6.26 Hasil Pengujian Integrasi *Method* `cekKolomIsian` *True*

Sedangkan untuk mendapatkan nilai *false* dari nilai kembalian *method* `cekKolomIsian`, akan dilakukan simulasi dengan mengosongi nilai kolom isian agar mendapat nilai *false*. Gambar 6.18 menunjukkan hasil integrasi antara *method* `buatProdukKhusus` dan *method* `cekKolomIsian` dengan nilai kembalian dari *method* `cekKolomIsian` bernilai *false*.

Gambar 6.27 Hasil Pengujian Integrasi *Method* `cekKolomIsian` *False*

Kemudian untuk mendapatkan nilai dari *method* `pesanProdukKhusus`, dilakukan simulasi dengan menginputkan nilai pada semua kolom isian agar mendapaktkan nilai *true* pada *method* `cekKolomIsian`, sehingga dapat mengeksekusi *method* `pesanProdukKhusus`. Gambar 6.19 menunjukkan hasil integrasi antara *method* `buatProdukKhusus` dan *method* `cekKolomIsian` dengan *method* `pesanProdukKhusus` yang diberi atribut status pemesanan untuk mengetahui status dari eksekusi *method* tersebut.

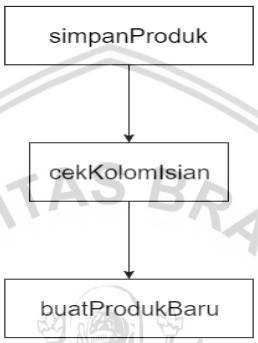


Gambar 6.28 Hasil Pengujian Integrasi *Method* `buatProdukKhusus`

Hasil dari penggunaan stub dan penggunaan *method* `buatProdukKhusus` yang sudah di integrasi kan dengan *method* `cekKolomIsian` dan *method* `pesanProdukKhusus` menghasilkan hasil yang sama. Sehingga dapat disimpulkan bahwa ketiga *method* tersebut telah berhasil terintegrasi.

6.1.3.2 Pengujian *Method* `simpanProduk()`

Method `simpanProduk` merupakan suatu *method* integrasi yang berfungsi untuk menyimpan produk sebagai syarat alur pembuatan produk baru. *Method* ini terintegrasi dengan *method* `cekKolomIsian` dan `buatProdukBaru` yang telah di uji pada pengujian unit. Diagram hiraki dari *method* yang saling terintegrasi tersebut di gambarkan oleh gambar 6.20 berikut :



Gambar 6.29 Diagram hiraki Pengujian Integrasi *Method* `simpanProduk`

Strategi pengujian integrasi yang akan digunakan untuk menguji *method* `cekKolomIsisan` adalah menggunakan pendekatan *Top Down*. Pada pendekatan ini pada *method* `simpanProduk` akan di buat sebuah *stub* yang berfungsi sebagai modul pengganti yang akan memanggil modul yang sedang di uji. Sedangkan untuk menguji *method* `buatProdukBaru`, hasil dari IS1 akan di buat sebuah *stub* yang berfungsi sebagai modul pengganti yang akan memanggil modul yang sedang di uji. Langkah – langkah dalam pengujian integrasi *method* `simpanProduk` seperti berikut ini :

Langkah 1 : IS1 = `simpanProduk + cekKolomIsian`

Langkah 2 : IS2 = IS1 + `buatProdukBaru`

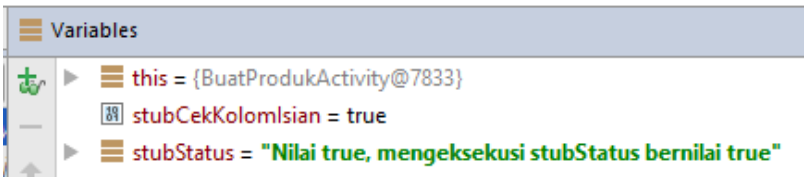
Pada langkah pertama pengujian *method* integrasi `simpanProduk` dan `cekKolomIsian` akan menggunakan *stub*. *Stub* yang akan digunakan pada pengujian *method* `cekKolomIsian` diberikan nilai *true* atau *false* yang merupakan tipe data boolean. Gambar 6.21 merupakan kode program *method* `simpanProduk` yang menggunakan stub dengan nilai *true* sebagai modul pengganti nilai kembalian *method* `cekKolomIsian`.

```
private void simpanProduk(){
    boolean stubCekKolomIsian = true;
    if (stubCekKolomIsian) {
        String stubStatus = "Nilai true, mengeksekusi
stubStatus bernilai true";
        namaProduk =
inputNamaProduk.getText().toString();
```

```
        hargaProduk =
Double.valueOf(inputHargaProduk.getText().toString())
;
        satuanProduk =
Integer.parseInt(inputNominalSatuan.getText().toStrin
g());
        deskripsiProduk =
inputDeskripsiProduk.getText().toString();
        long waktuDibuat = new Date().getTime();
        Log.d("nilai harga", "" + hargaProduk);

if(InternetConnection.getInstance().isOnline(BuatProd
ukActivity.this)) {
            try {
                if (listImage.size() == 0) {
                    ShowAlertDialog.showAlert("Anda
harus memilih gambar produk minimal (1)!", this);
                } else {
                    showProgessDialog();
                    ProdukModel produkModel = new
ProdukModel(getUid(), waktuDibuat, namaProduk,
kategoriProduk,hargaProduk, satuanProduk, namaSatuan,
namaAreaPenjual, deskripsiProduk);
produkPresenter.buatProdukBaru(produkModel,
listImage);
                }
            } catch (Exception e) {
                ShowSnackbar.showSnack(this,
getResources().getString(R.string.msg_error));
            }
        } else {
            final Snackbar snackbar =
Snackbar.make(activity,
getResources().getString(R.string.msg_noInternet),
Snackbar.LENGTH_INDEFINITE);
snackbar.setAction(getResources().getString(R.string.
ok), new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    snackbar.dismiss();
                }
            });
            snackbar.show();
        }
    } else {
        String stubStatus = "Nilai false,
mengeksekusi stubStatus bernilai false";
        ShowAlertDialog.showAlert("Anda harus mengisi
semua Form yang tersedia !", this);
    }
}
```

Gambar 6.30 Kode Program Stub *Method* `stubCekKolomIsian` *True*

Hasil dari penggunaan stub dengan nilai variabel `stubCekKolomIsian` *true* adalah *method* `simpanProduk` berhasil mengeksekusi nilai `String stubStatus` yang merupakan *statement* pada kondisi jika `stubCekKolomIsian` bernilai *true*. Hal ini ditunjukkan oleh Gambar 6.22.



Gambar 6.31 Hasil Pengujian Menggunakan Stub *Method* `stubCekKolomIsian` *True*

Selanjutnya adalah mengganti nilai `stubCekKolomIsian` dengan nilai *false* sebagai modul pengganti nilai kembalian *method* `cekKolomIsian`. Gambar 6.23 menunjukkan penggunaan *stub* dengan nilai *stub* sama dengan *false*.

```
private void simpanProduk(){
    boolean stubCekKolomIsian = false;
    if (stubCekKolomIsian) {
        String stubStatus = "Nilai true, mengeksekusi
stubStatus bernilai true";
        namaProduk =
inputNamaProduk.getText().toString();
        hargaProduk =
Double.valueOf(inputHargaProduk.getText().toString())
;
        satuanProduk =
Integer.parseInt(inputNominalSatuan.getText().toStrin
g());
        deskripsiProduk =
inputDeskripsiProduk.getText().toString();
        long waktuDibuat = new Date().getTime();
        Log.d("nilai harga", "" + hargaProduk);
if(InternetConnection.getInstance().isOnline(BuatProd
ukActivity.this)) {
            try {
                if (listImage.size() == 0) {
                    ShowAlertDialog.showAlert("Anda
harus memilih gambar produk minimal (1)!", this);
                } else {
                    showProgessDialog();
                    ProdukModel produkModel = new
ProdukModel(getUid(), waktuDibuat, namaProduk,
kategoriProduk,hargaProduk, satuanProduk, namaSatuan,
namaAreaPenjual, deskripsiProduk);
produkPresenter.buatProdukBaru(produkModel,
listImage);
                }
            } catch (Exception e) {
                ShowSnackbar.showSnack(this,
getResources().getString(R.string.msg_error));
            }
        } else {
            final Snackbar snackbar =
Snackbar.make(activity,
getResources().getString(R.string.msg_noInternet),
Snackbar.LENGTH_INDEFINITE);
snackbar.setAction(getResources().getString(R.string.
ok), new View.OnClickListener() {
                @Override
                public void onClick(View v) {
```

```
                    snackbar.dismiss();
                }
            });
            snackbar.show();
        }
    } else {
        String stubStatus = "Nilai false,
mengeksekusi stubStatus bernilai false";
        ShowAlertDialog.showAlert("Anda harus mengisi
semua Form yang tersedia !", this);
    }
}
```

Gambar 6.32 Kode Program Stub *Method* `stubCekKolomIsian` *False*

Hasil dari penggunaan *stub* dengan nilai variabel `stubCekKolomIsian` *false* adalah *method* `simpanProduk` berhasil mengeksekusi nilai `String stubStatus` yang merupakan statement pada kondisi jika `stubCekKolomIsian` bernilai *false*. Hal ini ditunjukkan oleh Gambar 6.24.

Gambar 6.33 Hasil Pengujian Menggunakan Stub *Method* `stubCekKolomIsian` *False*

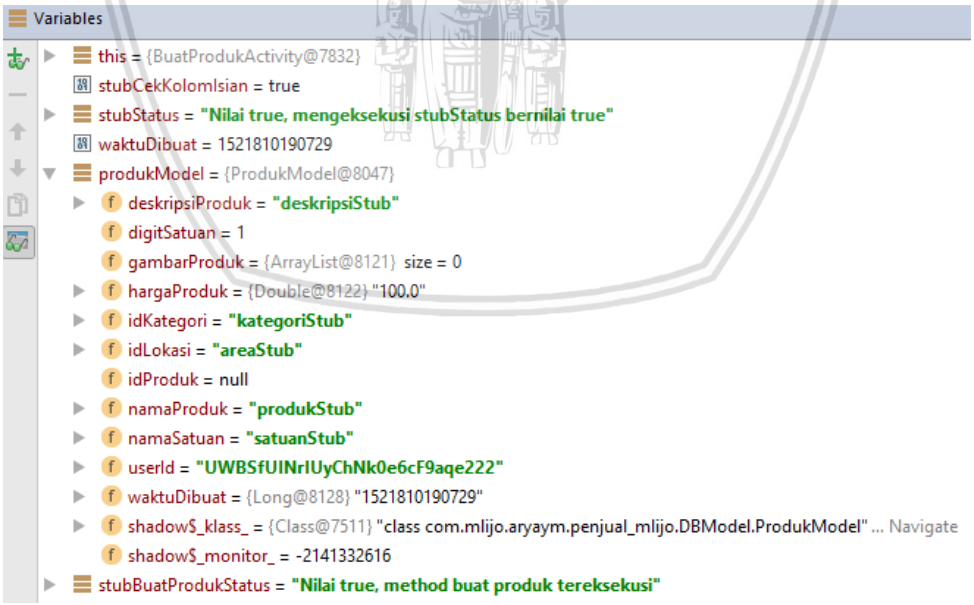
Kemudian dilanjutkan dengan pengujian integrasi langkah kedua. Dimana pada pengujian integrasi kedua *method* `simpanProduk` dan `cekKolomIsian` terintegrasi dengan *method* `buatProdukBaru`. Pada pengujian ini menggunakan pendekatan *Top Down* yang mana menggunkan *stub*. *Stub* yang digunakan pada pengujian *method* `simpanProduk` berupa *method dummy* `stubBuatProduk` yang diberikan nilai *dummy* dan fungsi seperti pada *method* `buatProduk`. Gambar 6.25 merupakan kode program *method* `simpanProduk` yang menggunakan stub *method* `stubBuatProdukBaru`.

```
private void simpanProduk(){
        boolean stubCekKolomIsian = true;
        if (stubCekKolomIsian) {
            String stubStatus = "Nilai true,
mengeksekusi stubStatus bernilai true";
            long waktuDibuat = new Date().getTime();
if(InternetConnection.getInstance().isOnline(BuatProd
ukActivity.this)) {
                try {
ProdukModel produkModel = new ProdukModel(getUid(),
waktuDibuat, "produkStub", "kategoriStub", 100, 1,
"satuanStub", "areaStub", "deskripsiStub");
produkPresenter.buatProdukBaru(produkModel,
listImage);
String stubBuatProdukStatus = "Nilai true, method
buat produk tereksekusi";
                } catch (Exception e) {
                    ShowSnackbar.showSnack(this,
```

```
getResources().getString(R.string.msg_error));
                    }
                } else {
                    final Snackbar snackbar =
Snackbar.make(activity,
getResources().getString(R.string.msg_noInternet),
Snackbar.LENGTH_INDEFINITE);

snackbar.setAction(getResources().getString(R.string.
ok), new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                            snackbar.dismiss();
                        }
                    });
                    snackbar.show();
                }
            } else {
                String stubStatus = "Nilai false,
mengeksekusi stubStatus bernilai false";
                ShowAlertDialog.showAlert("Anda harus
mengisi semua Form yang tersedia !", this);
            }
        }
```

Gambar 6.34 Kode Program Stub *Method* `stubBuatProdukBaru`

Hasil dari penggunaan *stub* dengan *method* `stubBuatProdukBaru` adalah *method* `simpanProduk` berhasil mengeksekusi *method* `stubBuatProdukBaru` dan dapat menjalankan fungsi seperti semestinya. Hal ini ditunjukkan oleh Gambar 6.26 dan 6.27.



Gambar 6.35 Hasil Submit stubBuatProdukBaru pada Console

Gambar 6.36 Hasil *Submit* `stubBuatProdukBaru` pada `Firebase`

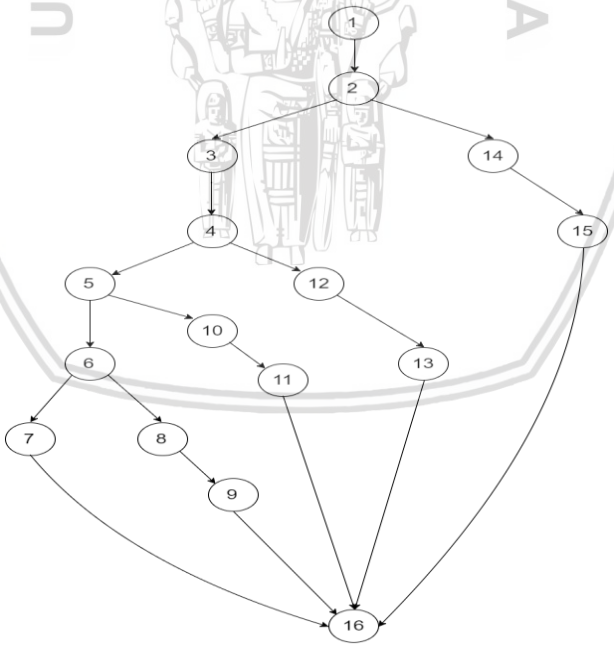
Selanjutnya adalah dengan mengintegrasikan *method* `simpanProduk` dengan *method* `cekKolomIsian` dan `buatProdukBaru`. Gambar 6.28 akan menjelaskan tentang kode program *method* `simpanProduk` secara keseluruan yang telah diintegrasikan dengan *method* `cekKolomIsian` dan `buatProdukBaru`.

```
private void simpanProduk(){
    if (cekKolomIsian() == true) {
namaProduk = inputNamaProduk.getText().toString();
hargaProduk=Double.valueOf(inputHargaProduk.getText()
.toString());
satuanProduk=Integer.parseInt(inputNominalSatuan
.getText().toString());
deskripsiProduk = inputDeskripsiProduk.getText()
.toString();
        long waktuDibuat = new Date().getTime();
if(InternetConnection.getInstance().isOnline(BuatProd
ukActivity.this)) {
            try {
                if (listImage.size() == 0) {
                    ShowAlertDialog.showAlert("Anda
harus memilih gambar produk minimal (1)!", this);
                } else {
                    showProgessDialog();
ProdukModel produkModel = new ProdukModel(getUid(),
waktuDibuat, namaProduk, kategoriProduk, hargaProduk,
satuanProduk, namaSatuan, namaAreaPenjual,
deskripsiProduk);
produkPresenter.buatProdukBaru(produkModel,
listImage);
String statusCekKolomIsian =
String.valueOf(cekKolomIsian());
String status = "cekKolomIsian true, buatProdukBaru tereksekusi";
                }
            } catch (Exception e) {
                ShowSnackbar.showSnack(this,
getResources().getString(R.string.msg_error));
            }
```

```
12          } else {
                final Snackbar snackbar =
Snackbar.make(activity,
getResources().getString(R.string.msg_noInternet),
Snackbar.LENGTH_INDEFINITE);
snackbar.setAction(getResources().getString(R.string.
ok), new View.OnClickListener() {                          13
                    @Override
                    public void onClick(View v) {
                        snackbar.dismiss();
                    }
                });
                snackbar.show();
            }
14      } else {
String statusCekKolomIsian =
String.valueOf(cekKolomIsian());
String status = "cekKolomIsian false, buatProdukBaru tidak tereksekusi";   15
        ShowAlertDialog.showAlert("Anda harus mengisi
semua Form yang tersedia !", this);
16      }
    }
```

Gambar 6.37 Kode Program *Method* `simpanProduk`

Berdasarkan proses pengujian tersebut, Gambar 6.29 menunjukkan *flow graph* dari *method* `simpanProduk` yang berfungsi menggambarkan alur dari *method* tersebut.



Gambar 6.38 *Flow Graph Method* `simpanProduk`

Tahapan yang akan dilakukan selanjutnya adalah menghitung *cyclomatic complexity*. Untuk menghitung *cyclomatic complexity* akan digunakan persamaan V(G) = E – N +2 dengan V(G) adalah jumlah kompleksitas siklomatis, lalu E (*edge*) adalah garis penghubung antar node dan N merupakan jumlah simpul (node).

V(G) = 6 – 6 + 2

= 0 + 2

= 2

Berdasarkan nilai *cyclomatic complexity* didapatkan dua buah basis set jalur *independent*, yaitu :

Jalur 1 : 1 – 2 – 3 – 4 – 5 – 6 – 7 – 16

Jalur 2 : 1 – 2 – 3 – 4 – 5 – 6 – 8 – 9 – 16

Jalur 3 : 1 – 2 – 3 – 4 – 5 – 10 – 11– 16

Jalur 4 : 1 – 2 – 3 – 4 – 12 – 13– 16

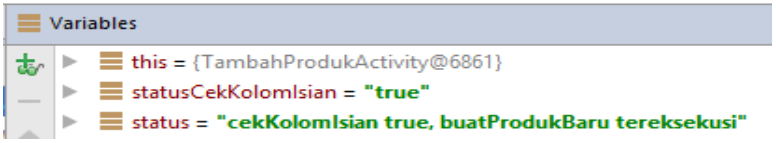
Jalur 5 : 1 – 2 – 14 – 15 – 16

Berdasarkan jumlah jalur yang didapatkan, maka akan dilakukan pengujian terhadap tiap jalur tersebut yang ditunjukkan oleh Tabel 6.5.

Tabel 6.23 Kasus Uji Pengujian Integrasi *Method* simpanProduk

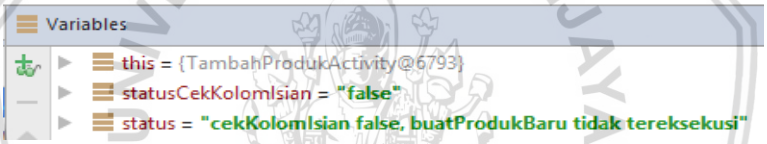
Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Ketika nilai *method* cekKolomIsian true, terkoneksi internet, tidak ada kesalahan sistem, dan belum memilih foto.	Menampilkan pesan peringatan untuk memilih foto minimal satu.	Menampilkan pesan peringatan untuk memilih foto minimal satu.
2	Ketika nilai *method* cekKolomIsian true, terkoneksi internet, tidak ada kesalahan sistem, telah pilih foto.	Menampilkan pesan peringatan untuk mengisi semua *form* yang tersedia. dan method buat produk baru sukses.	Menampilkan pesan peringatan untuk mengisi semua *form* yang tersedia.
3	Ketika nilai *method* cekKolomIsian true, terkoneksi internet, dan terdapat kesalahan sistem.	Menampilkan pesan *error*.	Menampilkan pesan *error*.
4	Ketika nilai *method* cekKolomIsian true, dan tidak terkoneksi internet.	Menampilkan pesan bahwa tidak terkoneksi internet.	Menampilkan pesan bahwa tidak terkoneksi internet.
5	Ketika nilai *method* cekKolomIsian false.	Menampilkan pesan bahwa form harus di isi semua.	Menampilkan pesan bahwa form harus di isi semua.

Selanjutnya adalah membandingkan nilai yang didapatkan sebelumnya menggunakan *stub* dengan nilai yang didapatkan dengan menggunakan *method* `cekKolomIsian` yang telah di integrasikan dengan *method* `simpanProduk`. Untuk mendapatkan nilai *true* dari nilai kembalian *method* `cekKolomIsian`, akan dilakukan simulasi dengan mengisi nilai kolom isian dengan suatu nilai agar mendapat nilai *true*. Gambar 6.30 menunjukkan hasil integrasi antara *method* `simpanProduk` dan *method* `cekKolomIsian` dengan nilai kembalian dari *method* `cekKolomIsian` bernilai *true*.



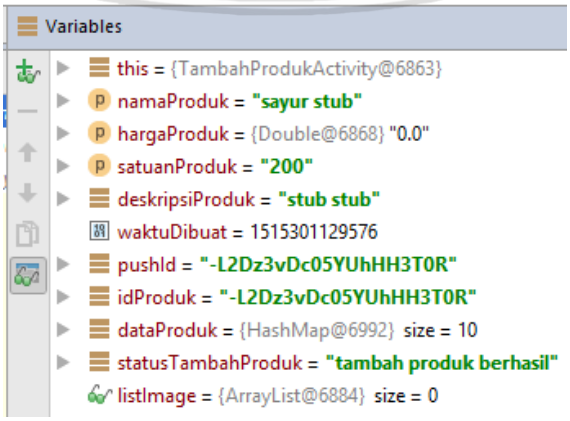
Gambar 6.39 Hasil Pengujian Integrasi *Method* cekKolomIsian True

Sedangkan untuk mendapatkan nilai *false* dari nilai kembalian *method* `cekKolomIsian`, akan dilakukan simulasi dengan mengosongi nilai kolom isian agar mendapat nilai *false*. Gambar 6.31 menunjukkan hasil integrasi antara *method* `simpanProduk` dan *method* `cekKolomIsian` dengan nilai kembalian dari *method* `cekKolomIsian` bernilai *false*.



Gambar 6.40 Hasil Pengujian Integrasi *Method* cekKolomIsian False

Kemudian untuk mendapatkan nilai dari *method* `buatProdukBaru`, dilakukan simulasi dengan menginputkan nilai pada semua kolom isian agar mendapaktkan nilai *true* pada *method* `cekKolomIsian`, sehingga dapat mengeksekusi *method* `buatProdukBaru`. Gambar 6.32 menunjukkan hasil integrasi antara *method* `simpanProduk` dan *method* `cekKolomIsian` dengan *method* `buatProdukBaru` yang diberi atribut status tambah produk untuk mengetahui status dari eksekusi *method* tersebut.



Gambar 6.41 Hasil Pengujian Integrasi *Method* `simpanProduk`

Hasil dari penggunaan stub dan penggunaan *method* `simpanProduk` yang sudah di integrasikan dengan *method* `cekKolomIsian` dan *method* `buatProdukBaru` menghasilkan hasil yang sama. Sehingga dapat disimpulkan bahwa ketiga *method* tersebut telah berhasil terintegrasi.

6.1.4 Pengujian Validasi

Pengujian validasi digunakan untuk dapat mengetahui apakah sistem yang dibangun sudah benar dan sesuai dengan daftar kebutuhan. Item - item yang telah dirumuskan dalam daftar kebutuhan dan merupakan hasil analisis kebutuhan akan menjadi acuan untuk melakukan pengujian validasi. Pengujian validasi menggunakan metode pengujian *Black Box*, karena tidak diperlukan konsentrasi terhadap alur jalannya algoritme program dan lebih ditekankan untuk menemukan perilaku antara kinerja sistem dengan daftar kebutuhan. Pada skripsi ini dilakukan pengujian validasi terhadap perangkat lunak Sistem Katalog dan Pemesanan Produk Kebutuhan Dapur Berbasis Android.

6.1.4.1 Kasus Uji Validasi Pengguna

a. Autentifikasi kondisi belum terdaftar

Tabel 6.24 Kasus Uji untuk pengujian validasi autentifikasi pengguna belum terdaftar

Nama Kasus Uji	Autentifikasi kondisi belum terdaftar
Tujuan Pengujian	Untuk membuktikan bahwa pengguna dapat melakukan pendaftaran kedalam sistem.
Prosedur Pengujian	1. Memasukkan nomor ponsel. 2. Menekan tombol selanjutnya. 3. Memasukkan nomor verifikasi. 4. Menekan tombol submit
Hasil yang Diharapkan	Sistem menampilkan halaman input data user baru.

b. Autentifikasi kondisi telah terdaftar

Tabel 6.25 Kasus Uji untuk pengujian validasi autentifikasi pengguna telah terdaftar

Nama Kasus Uji	Autentifikasi kondisi telah terdaftar
Tujuan Pengujian	Untuk membuktikan bahwa pengguna dapat melakukan login kedalam sistem.
Prosedur Pengujian	1. Memasukkan nomor ponsel. 2. Menekan tombol selanjutnya. 3. Memasukkan nomor verifikasi. 4. Menekan tombol submit
Hasil yang Diharapkan	Sistem menampilkan halaman *dashboard*.

c. Input Data User Baru

Tabel 6.26 Kasus Uji untuk pengujian validasi input data user baru

Nama Kasus Uji	Input Data User Baru
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menyimpan data user.
Prosedur Pengujian	1. Memasukkan data. 2. Menekan tombol simpan.
Hasil yang Diharapkan	Sistem akan menampilkan halaman *dashboard*.

6.1.4.2 Kasus Uji Validasi Konsumen

a. Melakukan pencarian produk berdasarkan kategori

Tabel 6.27 Kasus Uji untuk pengujian validasi pencarian produk berdasarkan kategori

Nama Kasus Uji	Melakukan pencarian produk berdasarkan kategori
Tujuan Pengujian	Untuk membuktikan bahwa konsumen dapat melakukan pencarian produk berdasarkan kategori.
Prosedur Pengujian	1. Memilih kategori produk pada halaman *dashboard*.
Hasil yang Diharapkan	Sistem menampilkan daftar produk dalam bentuk *listview* sesuai dengan kategori pencarian.

b. Melakukan pencarian dengan kata kunci

Tabel 6.28 Kasus Uji untuk pengujian validasi pencarian produk berdasarkan kata kunci

Nama Kasus Uji	Melakukan pencarian produk dengan kata kunci
Tujuan Pengujian	Untuk membuktikan bahwa konsumen dapat melakukan pencarian produk.
Prosedur Pengujian	1. Memilih icon pencarian pada halaman *dashboard*. 2. Memasukkan kata kunci. 3. Menekan tombol enter/cari.
Hasil yang Diharapkan	Sistem menampilkan produk dalam bentuk *listview* sesuai dengan kata kunci pencarian.

c. Melakukan filter pencarian

Tabel 6.29 Kasus Uji untuk pengujian validasi filter pencarian

Nama Kasus Uji	Melakukan filter pencarian
Tujuan Pengujian	Untuk membuktikan bahwa konsumen dapat melakukan filter pencarian pada sistem.
Prosedur Pengujian	1. Memilih menu filter pada daftar produk. 2. Memilih kategori filter. 3. Menekan tombol ya.
Hasil yang Diharapkan	Sistem menampilkan daftar produk dalam bentuk *listview* sesuai dengan filter pencarian.

d. Melihat daftar produk

Tabel 6.30 Kasus Uji untuk pengujian validasi daftar produk

Nama Kasus Uji	Melihat daftar produk
Tujuan Pengujian	Untuk membuktian bahwa sistem dapat menampilkan daftar produk.
Prosedur Pengujian	1. Memilih menu kategori
Hasil yang Diharapkan	Sistem menampilkan daftar produk dalam *listview*.

e. Menampilkan detail produk

Tabel 6.31 Kasus Uji untuk pengujian validasi detail produk

Nama Kasus Uji	Menampilkan detail produk
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menampilkan informasi detail produk.
Prosedur Pengujian	1. Memilih produk yang diinginkan dalam halaman daftar produk.
Hasil yang Diharapkan	Sistem menampilkan detail produk terpilih.

f. Melakukan pemesanan produk

Tabel 6.32 Kasus Uji untuk pengujian validasi melakukan pemesanan produk

Nama Kasus Uji	Melakukan pemesanan
Tujuan Pengujian	Untuk membuktikan bahwa konsumen dapat melakukan proses pemesanan produk pada sistem.

Prosedur Pengujian	1. Masukkan jumlah pesanan. 2. Menekan tombol pesan.
Hasil yang Diharapkan	Sistem melakukan proses pemesanan dan menyimpan data pesanan dalam database kemudian kembali ke halaman *dashboard*.

g. Melakukan pemesanan khusus

Tabel 6.33 Kasus Uji untuk pengujian validasi pemesanan produk khusus

Nama Kasus Uji	Melakukan pemesanan khusus
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat melakukan pemesanan produk khusus.
Prosedur Pengujian	1. Menekan tab penjual pada halaman *dashboard*. 2. Menekan tombol pesan produk khusus. 3. Mengisi informasi pesanan. 4. Menekan tombol pesan.
Hasil yang Diharapkan	Sistem melakukan proses pemesanan dan menyimpan data pesanan dalam database kemudian kembali ke halaman detail penjual.

h. Mengelola pembelian

Tabel 6.34 Kasus Uji untuk pengujian validasi mengelola pembelian

Nama Kasus Uji	Mengelola pembelian
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menampilkan menu mengelola pembelian
Prosedur Pengujian	1. Memilih menu drawer. 2. Memilih menu kelola pembelian.
Hasil yang Diharapkan	Sistem menampilkan halaman kelola pembelian.

i. Melihat detail pemesanan

Tabel 6.35 Kasus Uji untuk pengujian validasi melihat detail pemesanan

Nama Kasus Uji	Melihat detail pemesanan
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menampilkan detail informasi pesanan.

Prosedur Pengujian	1. Memilih salah satu menu pada halaman kelola pembelian. 2. Memlih pesanan yang ingin dilihat.
Hasil yang Diharapkan	Sistem menampilkan detail pesanan dalam bentuk halaman *activity*.

j. Menampilkan status pesanan

Tabel 6.36 Kasus Uji untuk pengujian validasi daftar produk

Nama Kasus Uji	Menampilkan status pesanan
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menampilkan status pesanan.
Prosedur Pengujian	1. Menekan menu status pesanan pada halaman kelola pembelian.
Hasil yang Diharapkan	Sistem menampilkan pesanan dalam bentuk *listview*.

k. Melakukan konfirmasi penerimaan

Tabel 6.37 Kasus Uji untuk pengujian validasi melakukan konfirmasi penerimaan

Nama Kasus Uji	Melakukan konfirmasi penerimaan
Tujuan Pengujian	Untuk membuktikan bahwa konsumen dapat melakukan proses konfirmasi penerimaan pada sistem.
Prosedur Pengujian	1. Memilih produk pesanan pada halaman status pesanan. 2. Menekan tombol terima.
Hasil yang Diharapkan	Sistem mengubah status pesanan menjadi diterima.

l. Memberikan Ulasan

Tabel 6.38 Kasus Uji untuk pengujian validasi memberikan ulasan

Nama Kasus Uji	Memberikan ulasan
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menyimpan ulasan.
Prosedur Pengujian	1. Memilih menu buat ulasan. 2. Mengisi kolom text input.

	3. Mengisi kolom penilaian bintang. 4. Menekan tombol kirim.
Hasil yang Diharapkan	Sistem menyimpan data ulasan dan kembali ke halaman daftar pesanan.

m. Melihat riwayat transaksi

Tabel 6.39 Kasus Uji untuk pengujian validasi melihat riwayat transaksi

Nama Kasus Uji	Melihat Riwayat Transaksi
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menampilkan riwayat transaksi.
Prosedur Pengujian	1. Memilih menu riwayat transaksi.
Hasil yang Diharapkan	Sistem menampilkan informasi riwayat transaksi di tampilkan dalam bentuk *listview* beserta status akhir.

n. Melihat info harga terkini

Tabel 6.40 Kasus Uji untuk pengujian validasi melihat info harga terkini

Nama Kasus Uji	Melihat informasi harga terkini
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menampilkan informasi harga terkini.
Prosedur Pengujian	1. Memilih menu info harga
Hasil yang Diharapkan	Sistem menampilkan informasi harga terkini di tampilkan dalam bentuk tabel

o. Melihat lokasi penjual aktif

Tabel 6.41 Kasus Uji untuk pengujian validasi melihat lokasi penjual aktif

Nama Kasus Uji	Melihat Lokasi Penjual Aktif
Tujuan Pengujian	Untuk menunjukkan bahwa sistem dapat melihat lokasi penjual yang sedang aktif.
Prosedur Pengujian	1. Memilih menu penjual aktif.
Hasil yang Diharapkan	Menampilkan lokasi penjual aktif ditampilkan dalam bentuk peta dan ikon yang dapat di klik.

p. Melihat lokasi penjual aktif kondisi gagal

Tabel 6.42 Kasus Uji untuk pengujian validasi melihat lokasi penjual aktif kondisi gagal

Nama Kasus Uji	Melihat Lokasi Penjual Aktif kondisi gagal
Tujuan Pengujian	Untuk menunjukkan bahwa sistem dapat memberikan pesan saat gagal mendapatkan lokasi penjual.
Prosedur Pengujian	1. Memilih menu penjual aktif.
Hasil yang Diharapkan	Menampilkan pesan tidak terdapat penjual di sekitar

q. Melihat Daftar penjual

Tabel 6.43 Kasus Uji untuk pengujian validasi melihat daftar penjual

Nama Kasus Uji	Melihat Daftar penjual
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menampilkan daftar penjual.
Prosedur Pengujian	1. Memilih tab penjual
Hasil yang Diharapkan	Menampilkan daftar penjual ditampilkan dalam bentuk *listview*.

r. Melihat profil penjual dari daftar penjual

Tabel 6.44 Kasus Uji untuk pengujian validasi melihat profil penjual dari daftar penjual

Nama Kasus Uji	Melihat profil penjual dari daftar penjual
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menampilkan detail informasi penjual
Prosedur Pengujian	1. Memilih salah satu penjual pada daftar penjual
Hasil yang Diharapkan	Sistem menampilkan profil penjual dalam bentuk activity.

s. Melihat profil penjual dari detail produk

Tabel 6.45 Kasus Uji untuk pengujian validasi melihat profil penjual dari detail produk

Nama Kasus Uji	Melihat profil penjual dari detail produk

Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menampilkan detail informasi penjual
Prosedur Pengujian	1. Menekan foto penjual pada halaman detail produk
Hasil yang Diharapkan	Sistem menampilkan profil penjual dalam bentuk activity.

t. Mengirimkan obrolan

Tabel 6.46 Kasus Uji untuk pengujian validasi mengirimkan obrolan

Nama Kasus Uji	Mengirimkan Obrolan
Tujuan Pengujian	Untuk membuktikan bahwa konsumen dapat mengirimkan obrolan kepada penjual
Prosedur Pengujian	1. Memilih tombol hubungi penjual 2. Input text pesan 3. Menekan tombol kirim
Hasil yang Diharapkan	Pesan terkirim kepada penjual dan sistem menampilkan pesan pada activity tersebut.

u. Mengelola data profil

Tabel 6.47 Kasus Uji untuk pengujian validasi mengelola data profil

Nama Kasus Uji	Mengelola data profil
Tujuan Pengujian	Untuk dapat mengelola data profil konsumen
Prosedur Pengujian	1. Memilih menu kelola profil 2. Mengisi input pada kolom yang disediakan. 3. Menekan tombol simpan.
Hasil yang Diharapkan	Sistem menyimpan profil konsumen yang diubah.

v. Logout

Tabel 6.48 Kasus Uji untuk pengujian validasi logout

Nama Kasus Uji	Logout
Tujuan Pengujian	Untuk membuktikan bahwa konsumen dapat keluar dari sistem.
Prosedur Pengujian	1. Memilih menu drawer. 2. Memilih menu logout.

Hasil yang Diharapkan	Sistem menampilkan halaman autentifikasi.

w. Melakukan pembatalan pemesanan

Tabel 6.49 Kasus Uji untuk pengujian validasi melakukan pembatalan pemesanan

Nama Kasus Uji	Melakukan Batal Pesan
Tujuan Pengujian	Untuk membuktikan bahwa konsumen dapat melakukan batal pesan
Prosedur Pengujian	1. Memilih pesanan baru. 2. Memilih pesanan yang dibatalkan. 3. Memlih menu batal pesan.
Hasil yang Diharapkan	Status pesanan berubah yang ditampilkan dalam bentuk *listview* dan berpindah ke daftar riwayat transaksi.

6.1.4.3 Kasus Uji Validasi Penjual

a. Mengelola data produk

Tabel 6.50 Kasus Uji untuk pengujian validasi mengelola data produk

Nama Kasus Uji	Mengelola data produk
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat masuk halaman kelola data produk yang dijual
Prosedur Pengujian	1. Menekan menu kelola produk
Hasil yang Diharapkan	Sistem menampilkan halaman kelola produk.

b. Membuat data produk

Tabel 6.51 Kasus Uji untuk pengujian validasi membuat data produk

Nama Kasus Uji	Membuat Data Produk
Tujuan Pengujian	Untuk dapat membuat data produk baru
Prosedur Pengujian	1. Menekan tombol tambah pada halaman kelola produk. 2. Mengisi form input. 3. Menekan tombol simpan.

Hasil yang Diharapkan	Sistem menyimpan input penjual dan menampilkan pada daftar kelola produk.

c. Merubah data produk

Tabel 6.52 Kasus Uji untuk pengujian validasi merubah data produk

Nama Kasus Uji	Merubah Data Produk
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menyimpan perubahan data produk
Prosedur Pengujian	1. Memilih icon config pada halaman kelola produk. 2. Memilih menu ubah. 3. Melakukan input data. 4. Menekan tombol simpan.
Hasil yang Diharapkan	Sistem menyimpan input penjual dan menampilkan produk pada daftar produk.

d. Menghapus data produk

Tabel 6.53 Kasus Uji untuk pengujian validasi menghapus data produk

Nama Kasus Uji	Menghapus data produk
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menghapus data produk dari daftar produk
Prosedur Pengujian	1. Memilih icon config pada halaman kelola produk. 2. Memilih menu hapus.
Hasil yang Diharapkan	Sistem menampilkan daftar produk pada halaman kelola produk.

e. Melihat detail produk

Tabel 6.54 Kasus Uji untuk pengujian validasi melihat detail produk

Nama Kasus Uji	Melihat detai produk
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menampilkan detail data produk
Prosedur Pengujian	1. Memilih produk pada daftar produk

Hasil yang Diharapkan	Sistem menampilkan detail produk pada halaman activity

f. Mengelola penjualan

Tabel 6.55 Kasus Uji untuk pengujian validasi mengelola penjualan

Nama Kasus Uji	Mengelola penjualan
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menampilkan halaman kelola penjualan produk
Prosedur Pengujian	1. Memilih menu kelola penjualan pada menu drawer
Hasil yang Diharapkan	Sistem menampilkan ringkasan status notifikasi pada halaman kelola penjualan.

g. Melihat daftar pesanan baru

Tabel 6.56 Kasus Uji untuk pengujian validasi melihat daftar pesanan baru

Nama Kasus Uji	Melihat daftar pesanan baru
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menampilkan halaman daftar pesanan baru
Prosedur Pengujian	1. Memilih menu pesanan baru pada halaman kelola penjualan
Hasil yang Diharapkan	Sistem menampilkan daftar pesanan baru dalam bentuk *listview*

h. Melihat informasi detail pesanan

Tabel 6.57 Kasus Uji untuk pengujian validasi melihat informasi detail pesanan

Nama Kasus Uji	Melihat informasi detail pesanan
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menampilkan halaman detail pesanan
Prosedur Pengujian	1. Memilih pesanan pada daftar pesanan
Hasil yang Diharapkan	Detail pesanan ditampilkan pada halaman *activity*

i. Melakukan konfirmasi pesanan

Tabel 6.58 Kasus Uji untuk pengujian validasi melakukan konfirmasi pesanan

Nama Kasus Uji	Melakukan konfirmasi pesanan
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat melakukan konfirmasi terhadap pesanan
Prosedur Pengujian	1. Memilih menu pesanan baru pada kelola penjualan 2. Memilih pesanan yang ingin di konfirmasi 3. Memilih tombol konfirmasi pada detail pesanan
Hasil yang Diharapkan	Pesanan terkonfirmasi dan status pesanan berubah.

j. Melihat peta menuju lokasi konsumen

Tabel 6.59 Kasus Uji untuk pengujian validasi melihat peta menuju lokasi konsumen

Nama Kasus Uji	Melihat peta menuju lokasi konsumen
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menampilkan peta lokasi konsumen
Prosedur Pengujian	1. Memilih lihat peta pada halaman detail pesanan
Hasil yang Diharapkan	Sistem menampilkan peta lokasi konsumen

k. Melihat status penjualan

Tabel 6.60 Kasus Uji untuk pengujian validasi melihat status penjualan

Nama Kasus Uji	Melihat status penjualan
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menampilkan status dari penjualan
Prosedur Pengujian	1. Memilih menu status pesanan pada kelola penjualan
Hasil yang Diharapkan	Sistem menampilkan daftar status pesanan dalam bentuk *listview*.

l. Memperbarui status penjualan

Tabel 6.61 Kasus Uji untuk pengujian validasi memperbarui status penjualan

Nama Kasus Uji	Memperbarui status penjualan

Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat melakukan perbaruan status dari pesanan
Prosedur Pengujian	1. Memilih menu status pesanan pada kelola penjualan 2. Memilih pesanan yang ingin di ubah statusnya 3. Memilih tombol perbarui status
Hasil yang Diharapkan	Status pesanan berubah

m. Melihat riwayat transaksi

Tabel 6.62 Kasus Uji untuk pengujian validasi melihat riwayat transaksi

Nama Kasus Uji	Melihat riwayat transaksi
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menampilkan riwayat dari transaksi penjualan
Prosedur Pengujian	1. Memilih menu riwayat transaksi pada kelola penjualan
Hasil yang Diharapkan	Sistem menampilkan daftar riwayat transaksi dalam bentuk *listview*

n. Mengelola status akun

Tabel 6.63 Kasus Uji untuk pengujian validasi mengelola status akun

Nama Kasus Uji	Mengelola status akun
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat melakukan kelola status akun penjual
Prosedur Pengujian	1. Memilih tombol switch aktif/non-aktif pada halaman *dashboard*
Hasil yang Diharapkan	Sistem akan merubah status akun penjual dan menampilkan informasi perubahan status

o. Mengelola data profil

Tabel 6.64 Kasus Uji untuk pengujian validasi mengelola data profil

Nama Kasus Uji	Mengelola data profil
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat melakukan kelola data profil penjual

Prosedur Pengujian	1. Memilih kelola profil pada halaman pengaturan 2. Melakukan perubahan 3. Memilih tombol simpan perubahan
Hasil yang Diharapkan	Data profil berubah dan berhasil diperbarui.

p. Melihat informasi harga terkini

Tabel 6.65 Kasus Uji untuk pengujian validasi melihat informasi harga terkini

Nama Kasus Uji	Melihat informasi harga terkini
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat menampilkan informasi harga terkini.
Prosedur Pengujian	1. Memilih menu info harga pada menu drawer
Hasil yang Diharapkan	Sistem menampilkan informasi harga terkini di tampilkan dalam bentuk tabel

q. Mengirimkan obrolan

Tabel 6.66 Kasus Uji untuk pengujian validasi mengirimkan obrolan

Nama Kasus Uji	Mengirimkan Obrolan
Tujuan Pengujian	Untuk membuktikan bahwa penjual dapat mengirimkan obrolan kepada konsumen
Prosedur Pengujian	1. Memilih tombol hubungi konsumen pada halaman detail pesanan 2. Input text pesan 3. Menekan tombol kirim
Hasil yang Diharapkan	Pesan terkirim kepada konsumen dan sistem menampilkan pesan pada activity tersebut.

r. Mengelola lokasi

Tabel 6.67 Kasus Uji untuk pengujian validasi mengelola lokasi

Nama Kasus Uji	Mengelola lokasi
Tujuan Pengujian	Untuk membuktikan bahwa sistem dapat melakukan kelola lokasi penjual
Prosedur Pengujian	1. Memilih menu pengaturan lokasi pada halaman pengaturan

	2. Menekan tombol switch aktif/non-aktif pada halaman Kelola lokasi berjualan
Hasil yang Diharapkan	Sistem akan merubah status lokasi penjual dan menampilkan informasi perubahan status

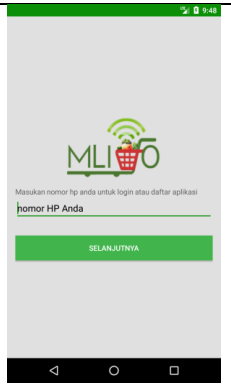
s. Logout

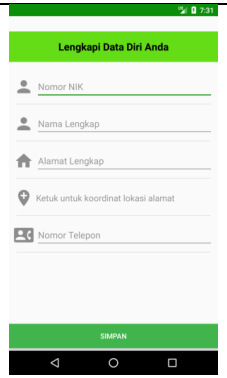
Tabel 6.68 Kasus Uji untuk pengujian validasi logout

Nama Kasus Uji	Logout
Tujuan Pengujian	Untuk membuktikan bahwa penjual dapat keluar dari sistem.
Prosedur Pengujian	1. Memilih menu drawer. 2. Memilih menu logout.
Hasil yang Diharapkan	Sistem menampilkan halaman autentifikasi.

6.1.4.4 Hasil Pengujian Validasi Pengguna

Tabel 6.69 Hasil pengujian Validasi Pengguna

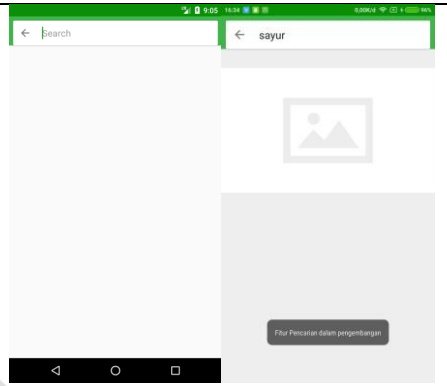
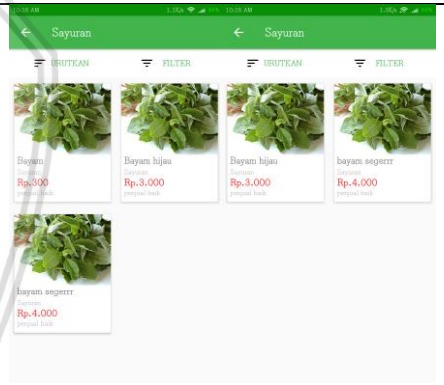
No.	Nama Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan	Screenshot Validasi	Status Validasi
1	Autentifikasi kondisi belum terdaftar	Sistem menampilkan halaman input data *user* baru.	Sistem menampilkan halaman input data *user* baru.		Valid
2	Autentifikasi kondisi telah terdaftar	Sistem menampilkan halaman *dashboard*.	Sistem menampilkan halaman *dashboard*.	Screenshot pada halaman ini sama seperti pada autentifikasi kondisi belum terdaftar, namun alurnya berbeda.	Valid

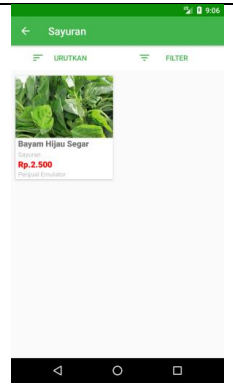
3	Input Data User Baru	Sistem akan menampilkan halaman *dashboard*.	Sistem akan menampilkan halaman *dashboard*.		Valid

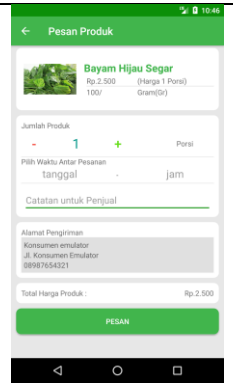
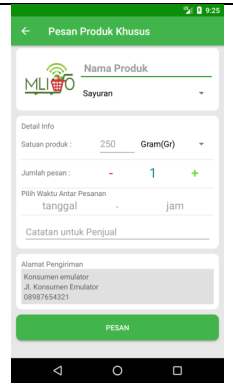
6.1.4.5 Hasil Pengujian Validasi Konsumen

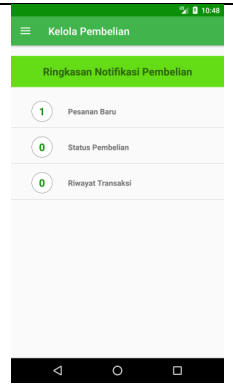
Tabel 6.70 Hasil Pengujian Validasi Konsumen

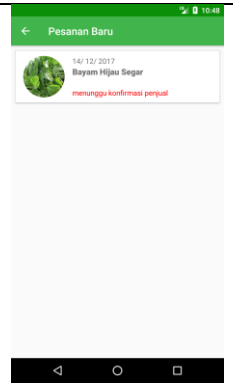
No.	Nama Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan	Screenshot Validasi	Status Validasi
1	Melakukan pencarian produk berdasarkan kategori	Sistem menampilkan daftar produk dalam bentuk *listview* sesuai dengan kategori pencarian.	Sistem menampilkan daftar produk dalam bentuk *listview* sesuai dengan kategori pencarian.		Valid

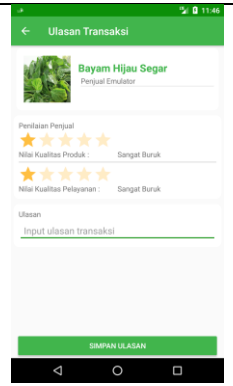
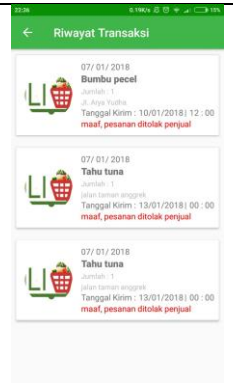
2	Melakukan pencarian produk dengan kata kunci	Sistem menampilkan produk dalam bentuk *listview* sesuai dengan kata kunci pencarian.	Fungsi pencarian tidak dapat dilakukan.		Invalid
3	Melakukan *filter* pencarian	Sistem menampilkan daftar produk dalam bentuk *listview* sesuai dengan *filter* pencarian.	Sistem menampilkan daftar produk dalam bentuk *listview* sesuai dengan *filter* pencarian.		Valid

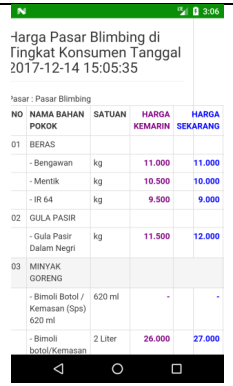
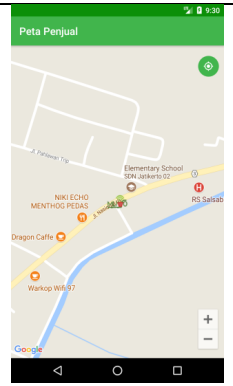
4	Melihat daftar produk	Sistem menampilkan daftar produk dalam *listview*.	Sistem menampilkan daftar produk dalam *listview*.		Valid
5	Menampilkan detail produk	Sistem menampilkan detail produk terpilih.	Sistem menampilkan detail produk terpilih.		Valid

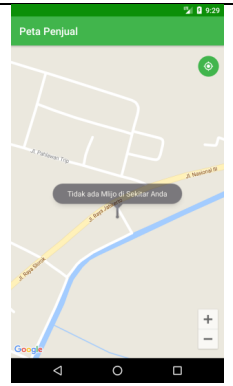
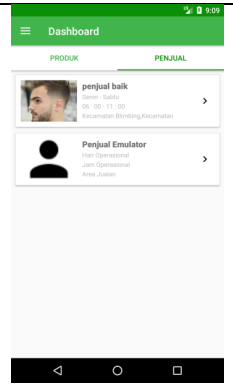
6	Melakukan pemesanan	Sistem melakukan proses pemesanan dan menyimpan data pesanan dalam database kemudian kembali ke halaman *dashboard*.	Sistem melakukan proses pemesanan dan menyimpan data pesanan dalam database kemudian kembali ke halaman *dashboard*.		Valid
7	Melakukan pemesanan khusus	Sistem melakukan proses pemesanan dan menyimpan data pesanan dalam database kemudian kembali ke halaman detail penjual.	Sistem melakukan proses pemesanan dan menyimpan data pesanan dalam database kemudian kembali ke halaman detail penjual.		Valid

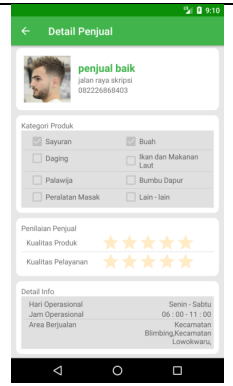
8	Mengelola pembelian	Sistem menampilkan halaman kelola pembelian.	Sistem menampilkan halaman kelola pembelian.		Valid
9	Melihat detail pemesanan	Sistem menampilkan detail pesanan dalam bentuk halaman *activity*.	Sistem menampilkan detail pesanan dalam bentuk halaman *activity*.		Valid

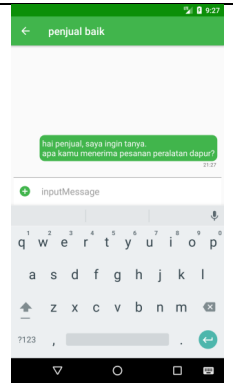
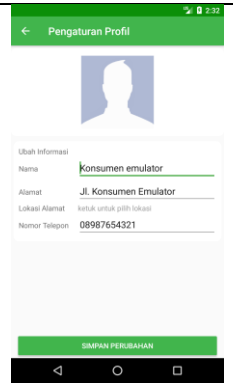
10	Menampilkan status pesanan	Sistem menampilkan pesanan dalam bentuk *listview*.	Sistem menampilkan pesanan dalam bentuk *listview*.		Valid
11	Melakukan konfirmasi penerimaan	Sistem mengubah status pesanan menjadi diterima.	Sistem mengubah status pesanan menjadi diterima.		Valid

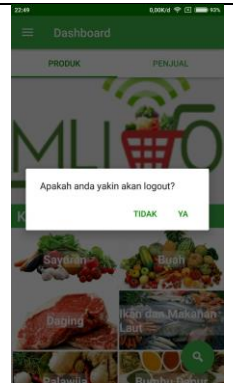
12	Memberikan ulasan	Sistem menyimpan data ulasan dan kembali ke halaman daftar pesanan.	Sistem menyimpan data ulasan dan kembali ke halaman daftar pesanan.		Valid
13	Melihat Riwayat Transaksi	Sistem menampilkan informasi riwayat transaksi di tampilkan dalam bentuk *listview* beserta status akhir.	Sistem menampilkan informasi riwayat transaksi di tampilkan dalam bentuk *listview* beserta status akhir.		Valid

14	Melihat informasi harga terkini	Sistem menampilkan informasi harga terkini di tampilkan dalam bentuk tabel	Sistem menampilkan informasi harga terkini di tampilkan dalam bentuk tabel		Valid
15	Melihat Lokasi Penjual Aktif	Menampilkan lokasi penjual aktif ditampilkan dalam bentuk peta dan ikon yang dapat di klik.	Menampilkan lokasi penjual aktif ditampilkan dalam bentuk peta dan ikon yang dapat di klik.		Valid

16	Melihat Lokasi Penjual Aktif kondisi gagal	Menampilkan pesan tidak terdapat penjual di sekitar	Menampilkan pesan tidak terdapat penjual di sekitar		Valid
17	Melihat Daftar penjual	Menampilkan daftar penjual ditampilkan dalam bentuk *listview*.	Menampilkan daftar penjual ditampilkan dalam bentuk *listview*.		Valid

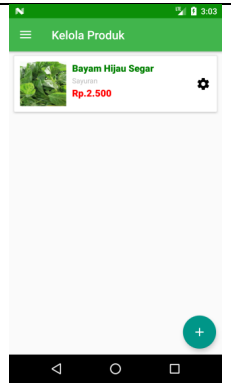
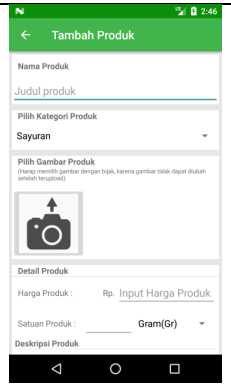
18	Melihat profil penjual dari daftar penjual	Sistem menampilkan profil penjual dalam bentuk activity.	Sistem menampilkan profil penjual dalam bentuk activity.		Valid
19	Melihat profil penjual dari detail produk	Sistem menampilkan profil penjual dalam bentuk activity.	Sistem menampilkan profil penjual dalam bentuk activity.		Valid

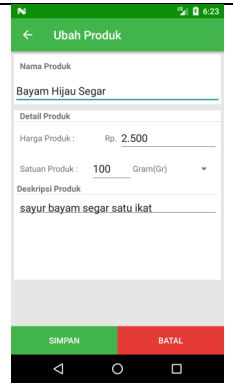
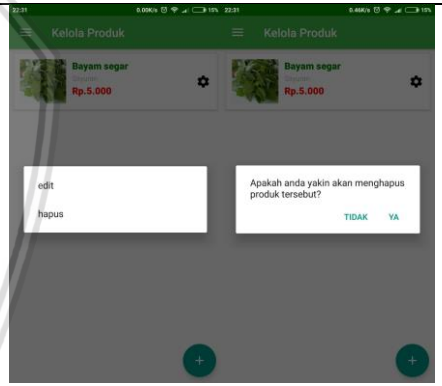
20	Mengirimkan Obrolan	Pesan terkirim kepada penjual dan sistem menampilkan pesan pada activity tersebut.	Pesan terkirim kepada penjual dan sistem menampilkan pesan pada activity tersebut.		Valid
21	Mengelola data profil	Sistem menyimpan profil konsumen yang diubah.	Sistem menyimpan profil konsumen yang diubah.		Valid

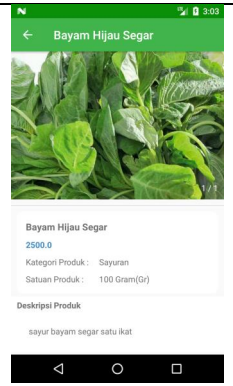
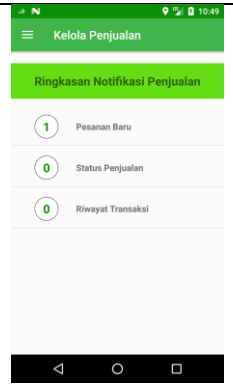
22	*Logout*	Sistem menampilkan halaman autentifikasi.	Sistem menampilkan halaman autentifikasi.		Valid
23	Melakukan Batal Pesan	Status pesanan berubah yang ditampilkan dalam bentuk *listview* dan berpindah ke daftar riwayat transaksi.	Status pesanan berubah yang ditampilkan dalam bentuk *listview* dan berpindah ke daftar riwayat transaksi.		Valid

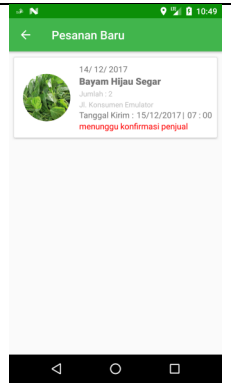
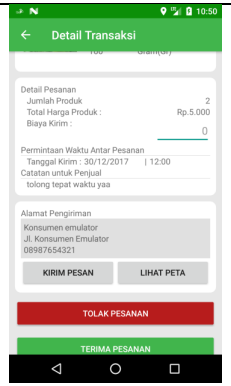
6.1.4.6 Hasil Pengujian Validasi Penjual

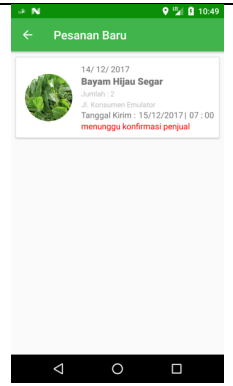
No.	Nama Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan	Screenshot Validasi	Status Validasi

1	Mengelola data produk	Sistem menampilkan halaman kelola produk.	Sistem menampilkan halaman kelola produk.		Valid
2	Membuat Data Produk	Sistem menyimpan input penjual dan menampilkan pada daftar kelola produk.	Sistem menyimpan input penjual dan menampilkan pada daftar kelola produk.		Valid

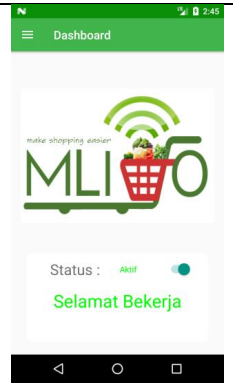
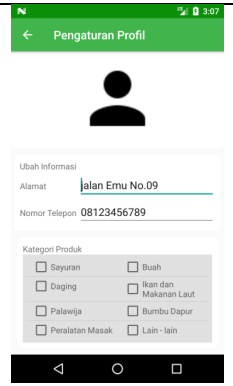
3	Merubah Data Produk	Sistem menyimpan input penjual dan menampilkan produk pada daftar produk.	Sistem menyimpan input penjual dan menampilkan produk pada daftar produk.		Valid
4	Menghapus data produk	Sistem menampilkan daftar produk pada halaman kelola produk.	Sistem menampilkan daftar produk pada halaman kelola produk.		Valid

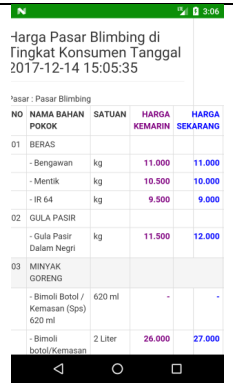
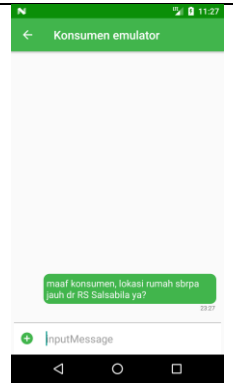
5	Melihat detai produk	Sistem menampilkan detail produk pada halaman activity	Sistem menampilkan detail produk pada halaman activity		Valid
6	Mengelola penjualan	Sistem menampilkan ringkasan status notifikasi pada halaman kelola penjualan.	Sistem menampilkan ringkasan status notifikasi pada halaman kelola penjualan.		Valid

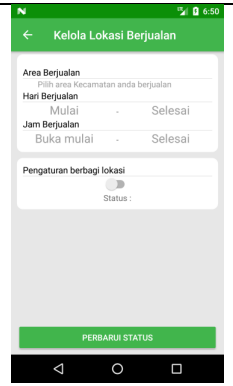
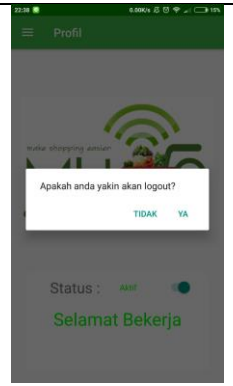
7	Melihat daftar pesanan baru	Sistem menampilkan daftar pesanan baru dalam bentuk *listview*	Sistem menampilkan daftar pesanan baru dalam bentuk *listview*		Valid
8	Melihat informasi detail pesanan	Detail pesanan ditampilkan pada halaman *activity*	Detail pesanan ditampilkan pada halaman *activity*		Valid
9	Melakukan konfirmasi pesanan	Pesanan terkonfirmasi dan status pesanan berubah.	Pesanan terkonfirmasi dan status pesanan berubah.	Screenshot pada halaman ini sama seperti pada halaman melihat informasi detail pesanan	Valid

10	Melihat peta menuju lokasi konsumen	Sistem menampilkan peta lokasi konsumen	Sistem menampilkan peta lokasi konsumen		Valid
11	Melihat status penjualan	Sistem menampilkan daftar status pesanan dalam bentuk *listview*.	Sistem menampilkan daftar status pesanan dalam bentuk *listview*.		Valid

12	Memperbarui status penjualan	Status pesanan berubah	Status pesanan berubah		Valid
13	Melihat riwayat transaksi	Sistem menampilkan daftar riwayat transaksi dalam bentuk *listview*	Sistem menampilkan daftar riwayat transaksi dalam bentuk *listview*		Valid

14	Mengelola status akun	Sistem akan merubah status akun penjual dan menampilkan informasi perubahan status	Sistem akan merubah status akun penjual dan menampilkan informasi perubahan status		Valid
15	Mengelola data profil	Data profil berubah dan berhasil diperbarui.	Data profil berubah dan berhasil diperbarui.		Valid

16	Melihat informasi harga terkini	Sistem menampilkan informasi harga terkini di tampilkan dalam bentuk tabel	Sistem menampilkan informasi harga terkini di tampilkan dalam bentuk tabel		Valid
17	Mengirimkan Obrolan	Pesan terkirim kepada konsumen dan sistem menampilkan pesan pada activity tersebut.	Pesan terkirim kepada konsumen dan sistem menampilkan pesan pada activity tersebut.		Valid

18	Mengelola lokasi	Sistem akan merubah status lokasi penjual dan menampilkan informasi perubahan status	Sistem akan merubah status lokasi penjual dan menampilkan informasi perubahan status		Valid
19	*Logout*	Sistem menampilkan halaman autentifikasi.	Sistem menampilkan halaman autentifikasi.		Valid

6.2 Pengujian Non-fungsional

Pada penelitian ini akan dilakukan pengujian terhadap kebutuhan non-fungsional. Pengujian kebutuhan non-fungsional yang akan dilakukan adalah pengujian *usability* dan pengujian *compability.*

6.2.1 Pengujian *Usability*

Pengujian *usability* merupakan pengujian yang dilakukan untuk mengetahui tanggapan atau kepuasan pengguna ketika menggunakan aplikasi yang telah dikembangkan. Dengan pengujian ini akan didapatkan suatu hasil yang dapat dijadikan tolak ukur apakah aplikasi yang sudah dibuat mudah digunakan atau tidak. Pada penelitian ini hanya akan mengukur tingkat kepuasan pengguna atau satisfaction untuk dapat menjawab rumusan masalah yang telah di definisikan sebelumnya. Untuk mengukur tingkat kepuasan pengguna tersebut, akan digunakan metode *System Usability Scale* (SUS) yang merupakan metode dari pengujian *usability*.

6.2.1.1 Identifikasi Pengguna

Pada penelitian ini pengujian *usability* akan ditujukan kepada 5 orang responden yang telah dijelaskan sebelumnya bawah menurut Nielsen (2000), tidak harus membutuhkan pengujian pengguna yang besar berkaitan dengan pemborosan sumber daya (Nielsen, 2000). Kelima responden tersebut juga sebelumnya telah ikut serta dalam proses iterasi untuk mengetahui tanggapan dari hasil akhir pengembangan sistem katalog dan pemesanan produk kebutuhan dapur di Kota Malang. Dari kelima responden tersebut akan dibagi menjadi dua kelompok. Kelompok pertama terdiri dari 4 responden bagi sisi konsumen dan 1 responden bagi sisi penjual. Jumlah responden konsumen mendominasi dikarenakan mayoritas fungsi dan tampilan di buat berdasarkan penilaian sisi konsumen.

6.2.1.2 System *Usability* Scale

a. Skenario Pengujian

Skenario pengujian pada pengujian usability dengan menggunakan SUS yaitu dengan memberikan pertanyaan kepada pengguna sesuai dengan 10 kriteria pertanyaan yang terdapat pada item SUS. Kriteria pertanyaan tersebut terbagi menjadi nomor ganjil dan genap. Nomor ganjil akan memuat pertanyaan yang bersifat positif dan pertanyaan genap memuat pertanyaan bersifat negatif. Pertanyaan-pertanyaan ini akan dijawab pengguna setelah selesai dan merasa cukup mencoba aplikasi. Pertanyaan ini juga berlaku pada kedua sisi yaitu sisi konsumen dan sisi penjual. Tabel 6.59 akan menjelaskan daftar pertanyaan untuk sisi konsumen dan Tabel 6.60 untuk sisi penjual.

Tabel 6.71 Daftar Pernyataan Kuisioner SUS pada Konsumen

No	Pernyataan Kuisioner
1	Saya akan menggunakan Sistem Pemesanan Kebutuhan Dapur dan Sayuran di Kota Malang(MLIJO) untuk melakukan pemesanan produk kebutuhan dapur.
2	Saya menilai sistem ini menyediakan alur yang kompleks.
3	Saya menilai sistem ini mudah untuk digunakan.
4	Saya membutuhkan bantuan orang lain untuk menggunakan sistem ini.
5	Saya menilai sistem ini menyediakan fungsi yang saling berintegrasi dengan baik.
6	Saya menilai sistem ini mengandung banyak hal yang tidak konsisten.
7	Saya merasa banyak orang yang akan dengan mudah menggunakan sistem ini untuk melakukan pemesanan produk kebutuhan dapur.
8	Saya menilai sistem ini sangat rumit untuk digunakan.
9	Saya merasa dapat menggunakan sistem ini dengan baik.
10	Saya butuh belajar lebih banyak untuk menggunakan sistem ini dengan baik.

Tabel 6.72 Daftar Pernyataan Kuisioner SUS pada Penjual

No	Pernyataan Kuisioner
1	Saya akan menggunakan Sistem Pemesanan Kebutuhan Dapur dan Sayuran di Kota Malang(MLIJO) untuk melakukan penjualan produk saya.
2	Saya menilai sistem ini menyediakan alur yang kompleks.
3	Saya menilai sistem ini mudah untuk digunakan.
4	Saya membutuhkan bantuan orang lain untuk menggunakan sistem ini.
5	Saya menilai sistem ini menyediakan fungsi yang saling berintegrasi dengan baik.
6	Saya menilai sistem ini mengandung banyak hal yang tidak konsisten.
7	Saya merasa banyak orang yang akan dengan mudah menggunakan sistem ini untuk melakukan penjualan produk kebutuhan dapur.
8	Saya menilai sistem ini sangat rumit untuk digunakan.
9	Saya merasa dapat menggunakan sistem ini dengan baik.
10	Saya butuh belajar lebih banyak untuk menggunakan sistem ini dengan baik.

b. Hasil Pengujian

Hasil pengujian *usability* dengan menggunakan metode SUS dari sisi konsumen dan penjual akan dikumpulkan untuk dihitung nilai skor sesuai dengan perhitungan metode SUS. Lalu skor tersebut dapat menjadi tolak ukur apakah sistem yang telah dikembangkan memilik tingkat kepuasan dan kemudahan yang tinggi atau tidak. Penilaian dari metode SUS adalah untuk item nomor ganjil 1, 3, 5, 7 dan 9 memiliki skor kontribusi posisi skala dikurangi dengan 1. Sedangkan untuk item nomor genap 2, 4, 6, 8 dan 10 skor kontribusinya adalah 5 dikurangi dengan nilai skala yang didapatkan pada tiap item genap. Kemudian dilakukan penjumlahan dari setiap skor item lalu dikalikan dengan 2,5 sehingga akan menghasilkan skor SUS. Selanjutnya total skor SUS dibagi dengan jumlah responden, sehingga didapatkan nilai rata-rata skor SUS.

Tabel 6.73 Hasil Perhitungan Pengujian SUS pada Konsumen

Responden	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Skor SUS
K1	5	2	5	2	3	4	4	2	4	2	72.5
K2	4	1	5	1	4	1	5	1	5	1	95
K3	5	2	5	1	4	1	5	1	5	1	95
K4	5	3	4	3	4	1	5	2	4	3	75
Total Skor SUS											337.5
Rata-Rata Skor SUS											84.37

Tabel 6.74 Hasil Perhitungan Pengujian SUS pada Penjual

Responden	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Skor SUS
P1	4	2	4	2	3	2	4	2	4	2	72.5
Total Skor SUS											72.5
Rata-Rata Skor SUS											72.5

6.2.2 Pengujian *Compability*

Pengujian *compability* adalah salah satu aktivitas pengujian sistem untuk aplikasi *mobile*, yang bertujuan untuk memvalidasi ketergantungan antara aplikasi *mobile* yang diuji dan lingkungan operasinya yang berbeda. Pengujian kompatibilitas *mobile* berfokus pada tiga masalah kompatibilitas utama yaitu,

kompatibilitas *platform*, kompatibilitas fitur perangkat, dan kompatibilitas *native API*. Dengan pengujian ini akan didapatkan suatu hasil yang dapat dijadikan tolak ukur apakah aplikasi yang telah dihasilkan dapat dijalankan pada beberapa system operasi Android yang berbeda. Pada penelitian ini akan menggunakan firebase Test Lab untuk melakukan uji kompatibilitas *native API*. Berikut hasil dari pengujian kompatibilitas *native API*. Yang ditunjukan oleh tabel 6.57 pada sisi aplikasi konsumen dan tabel 6.58 pada sisi aplikasi penjual.

Tabel 6.75 Hasil pengujian *Compability* aplikasi Konsumen

No	Perangkat Uji	Hasil Uji
1	Nexus 5, Virtual, API Level 19	sukses
2	Nexus 5, Virtual, API Level 21	sukses
3	Nexus 5, Virtual, API Level 22	sukses
4	Nexus 5, Virtual, API Level 23	sukses
5	Nexus 5X, Virtual, API Level 24	sukses
6	Nexus 5X, Virtual, API Level 25	sukses
7	Nexus 5X, Virtual, API Level 26	sukses

Tabel 6.76 Hasil pengujian *Compability* aplikasi Penjual

No	Perangkat Uji	Hasil Uji
1	Nexus 5, Virtual, API Level 19	sukses
2	Nexus 5, Virtual, API Level 21	sukses
3	Nexus 5, Virtual, API Level 22	sukses
4	Nexus 5, Virtual, API Level 23	sukses
5	Nexus 5X, Virtual, API Level 24	sukses
6	Nexus 5X, Virtual, API Level 25	sukses
7	Nexus 5X, Virtual, API Level 26	sukses

6.3 Analisis Hasil Pengujian

Tahap analisis hasil pengujian bertujuan untuk mendapatkan kesimpulan dari hasil pengujian sistem katalog dan pemesanan produk kebutuhan dapur berbasis android. Analisis dilakukan terhadap semua hasil pengujian yang telah dilakukan pada penelitian ini, yang meliputi analisis hasil pengujian unit, pengujian integrasi, pengujian validasi dan pengujian *usability*.

6.3.1 Analisis Hasil *Test Driven Development*

Analisis hasil test driven development dilakukan untuk mengetahui apakah pengujian yang dilakukan pada algoritme yang telah di rancang sebelumnya dapat

berjalan sesuai dengan skenario yang telah di definisikan. Berdasarkan pengujian yang telah dilakukan, tdd terhadap 3 unit algoritme pada penelitian ini dapat disimpulkan jika 3 unit algoritme tersebut telah sesuai dengan skenario yang dirancang sebelumnya, sehingga dapat di implementasikan kedalam kode program utama.

6.3.2 Analisis Hasil Pengujian Unit

Analisis hasil pengujian unit dilakukan untuk mengetahui apakah unit modul yang telah diuji memiliki kesamaan atau kesesuaian dengan perancangan yang telah dirancangan sebelumnya. Berdasarkan pengujian yang telah dilakukan, terdapat 3 pengujian unit yang dilakukan pada penelitian ini dan dapat disimpulkan 3 modul unit yang di uji tersebut sudah memenuhi kebutuhan fungsional yang telah dirancang pada bagian perancangan.

6.3.3 Analisis Hasil Pengujian Integrasi

Analisis hasil dari pengujian integrasi dilakukan dengan cara melihat apakah fungsi dari sistem atau kebutuhan fungsional sistem yang telah diimplementasikan dalam unit modul dapat saling berintegrasi dengan baik sesuai perancangan sistem yang sebelumnya telah dirancang. Berdasarkan hal tersebut dalam disimpulkan bahwa *method* integrasi yang terdiri dari beberapa unit modul dari program sudah memenuhi kebutuhan fungsional seperti yang telah dirancangan pada tahap perancangan.

6.3.4 Analisis Hasil Pengujian Validasi

Analisis hasil dari pengujian validasi dilakukan dengan cara membandingkan antara hasil uji dengan daftar kebutuhan yang telah dituliskan sebelumnya. Jika hasil uji mempunyai kesesuaian dengan skenario sistem, maka sistem tersebut adalah valid karena telah mengimplementasikan fungsi atau kebutuhan fungsional. Apabila menghasilkan hasil uji yang tidak valid, maka sistem tersebut belum memenuhi semua kebutuhan fungsional. Berdasarkan hal tersebut dapat disimpulkan bahwa hasil pengujian validasi Sistem Katalog dan Pemesanan Produk Kebutuhan Dapur Berbasis Android telah memenuhi kebutuhan fungsional yang telah di definisikan sebelumya karena mayoritas kebutuhan fungsional yang telah di uji menghasilkan hasil uji yang valid. Sedangkan pada pengujian validasi konsumen pencarian produk berdasarkan kata kunci penuh mendapatkan hasil invalid dikarenakan firebase sebagai layanan *web service* yang digunakan tidak memungkinkan untuk dapat berkomunikasi dengan *library* pihak ketiga apabila menggunakan layanan tipe *spark*(tidak berbayar). Yang mana pada penelitian ini fungsi pencarian dengan kata kunci menggunakan *library* pihak ketiga yaitu `algolia` untuk dapat menjalankan fungsi tersebut, sehingga fungsi tersebut tidak dapat berjalan dengan semestinya dan dinyatakan invalid.

6.3.5 Analisis Hasil Pengujian *Usability*

Analisis hasil dari pengujian *usability* dengan menggunakan metode SUS dilakukan dengan melihat *range* penilaian untuk menentukan apakah sistem yang

telah dikembangkan diterima oleh pengguna atau tidak. Skor SUS dengan rentang penilaian 0 – 50 termasuk kedalam kategori "*Not Acceptable*", skor SUS 51 – 70 termasuk dalam kategori "*Marginal*" dan skor SUS dengan rentang 71 – 100 termasuk dalam kategori "*Acceptable*". Pada sisi konsumen menghasilkan nilai SUS 84.37 yang mana nilai tersebut dihasilkan dari penilaian positif pengguna pada pertanyaan 1,3,7, dan 9 yang menyatakan akan kemudahan pada penggunaan aplikasi sehingga dapat digunakan dengan baik. Sedangkan pada sisi penjual menghasilkan nilai SUS 72,5 yang mana nilai tersebut dihasilkan dari penilaian positif pengguna pada pertanyaan 1,3,7, dan 9 yang menyatakan akan kemudahan pada penggunaan aplikasi sehingga dapat digunakan dengan baik. Dengan nilai rata-rata skor SUS sisi konsumen 84.37 dan pada sisi penjual 72,5, maka dapat disimpulkan bahwa Sistem Katalog Dan Pemesanan Produk Kebutuhan Dapur Berbasis Android mudah digunakan dan dapat diterima oleh pengguna.

6.3.6 Analisis Hasil Pengujian *Compability*

Analisis hasil dari pengujian compability dilakukan dengan cara melihat hasil dari run test firebase Test Lab, apakah berjalan sukses atau terdapat crash. Berdasarkan pengujian yang telah dilakukan kepada kedua aplikasi konsumen dan penjual pada perangkat Uji Android API 19 (kitkat) hingga 26 (oreo), dihasilkan hasil uji dengan status sukses. Sehingga dapat di simpulkan bahwa pengujian compability berhasil dijalankan pada semua perangkat android yang telah di tentukan pada tahap analisis kebutuhan bagian pemilihan lingkungan pengembangan bab 4.1.3.

BAB 7 PENUTUP

Bagian ini merupakan bagian akhir laporan. Bagian ini berisi dua sub bab, yaitu kesimpulan dan saran dari penulis.

7.1 Kesimpulan

Berdasarkan analisis yang telah dilakukan dari penelitian tersebut, dapat diambil kesimpulan diantaranya :

1. Pengembangan sistem katalog dan pemesanan produk kebutuhan dapur berbasis android menggunakan metode *Mobile*-d pada tahap analisis kebutuhan dapat menghasilkan 43 daftar kebutuhan yang jelas dan detail dimana memungkinkan untuk dilakukan penggalian kebutuhan berulang – ulang sesuai dengan siklus iterasi yang ada pada alur metode tersebut seperti yang di tunjukan pada analisis kebutuhan pada bab 4.
2. Pengembangan sistem katalog dan pemesanan produk kebutuhan dapur berbasis android menggunakan metode *Mobile*-d pada tahap perancangan dapat menghasilkan beberapa rancangan, diantaranya perancangan arsitektural, basis data, diagram *class*, diagram *sequence*, algoritme, dan antarmuka. Pada perancangan basis data di hasilkan skema basis data yang menggambarkan struktur data pada *firebase database* yang berbasis JSON. Pada perancangan diagram *class*, di hasilkan rancangan diagram yang jelas sesuai hasil iterasi yeng telah dilakukan. Sedangkan pada perancangan algoritme dihasilkan alur algoritme sistem yang digunakan sebagai dasar implementasi kode program, dimana tahap ini merupakan adopsi dari metode Mobile-d *task* TDD.
3. Pengembangan sistem katalog dan pemesanan produk kebutuhan dapur berbasis android menggunakan metode *Mobile*-d pada tahap implementasi dapat menghasilkan beberapa hasil implementasi, berupa implementasi basis data, implementasi class, kode program dan juga antarmuka. Pada imlementasi kode program, penggunaan *realtime database* dari *firebase* tidak dapat diimplementasikan pada fungsi sistem yang menggunakan kondisi *multiple where clause*. Hal ini di tunjukan pada implementasi fungsi *filter* pencarian produk yang tidak dapat berjalan dengan semestinya dikarenakan hanya dapat melakukan *query* pada satu kondisi saja, sehingga digunakan *firebase firestore* untuk menyimpan data produk.
4. Pengembangan sistem katalog dan pemesanan produk kebutuhan dapur berbasis android menggunakan metode *Mobile*-d pada tahap pengujian dapat menghasilkan pengujian fungsional dan non-fungsional. Pada pengujian fungsional dilakukan pengujian unit, integrasi, dan validasi, dimana pada pengujian unit berhasil melakukan

semua jalur uji, sedangkan pada pengujian integrasi *method – method* terkait dapat terintegrasi sesuai dengan fungsinya. Pada pengujian validasi terhadap 45 fungsi menghasilkan 1 fungsi invalid, yang disebabkan fungsi tersebut tidak dapat di implementasikan karena keterbatasan *environment* seperti yang telah di jelaskan pada sub-bab analisis hasil pengujian validasi.

5. Pengembangan sistem katalog dan pemesanan produk kebutuhan dapur berbasis android menggunakan metode *Mobile*-d pada tahap pengujian *usability* menggunakan teknik SUS mendapatkan nilai 84,37 pada sisi konsumen dan 72,5 pada sisi penjual dari nilai maksimum penilaian 100. Sehingga dapat disimpulkan bahwa aplikasi ini dapat diterima dan di gunakan dengan mudah oleh pengguna.

7.2 Saran

Berdasarkan hasil penelitian dan kesimpulan yang telah di tuliskan sebelumnya, maka penulis dapat menuliskan saran untuk penelitian selanjutnya diantaranya sebagai berikut:

1. Pada fungsi sistem sebaiknya ditambahkan fungsi hitung waktu konfirmasi secara otomatis, sehingga dapat lebih mempermudah informasi bagi konsumen saat menunggu konfirmasi dari penjual.
2. Pada sisi penjual dapat di tambahkan fungsi pengaturan waktu pengiriman, sehingga tidak terdapat konsumen yang memesan pada waktu yang tidak mencakup jam kerja penjual.
3. Penelitian selanjutnya juga dapat menambahkan fungsi analisis penjualan pada sisi penjual, seperti misalnya produk yang paling sering di pesan.
4. Penelitian selanjutnya dapat menggunakan tipe *database* yang lain, sehingga fungsi pencarian dengan menggunakan kata kunci penuh dapat berjalan dengan semestinya.
5. Dapat menggunakan metode pengembangan aplikasi *mobile* yang lain. Sehingga dapat digunakan sebagai bahan perbandingan dengan penelitian yang telah dilakukan.

DAFTAR PUSTAKA

Abrahamsson, P. et al., 2004. Mobile-D: An Agile approach for mobile application development. *In proceedings of OOPSLA.*

Alluri, A., 2012. *Usability Testing of Android Applications.* San Diego: s.n.

Ambler, S., 2013. *Introduction to Test Driven Development (TDD).* [Online] Available at: http://agiledata.org/essays/tdd.html [Diakses 04 Desember 2017].

APJII, 2016. *Penetrasi dan Perilaku Pengguna Internet Indonesia,* Jakarta: Asosiasi Penyelenggara Jasa Internet Indonesia.

Badan Pusat Statistik, 2016. *Penduduk Berumur 15 Tahun ke Atas yang Bekerja Menurut Kabupaten/Kota dan Lapangan Pekerjaan Utama, Tahun 2015.* [Online] Available at: https://jatim.bps.go.id/LinkTabelStatis/view/id/358 [Diakses 6 April 2017].

Bangor, A., Kortum, P. & Miller, J., 2009. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *Journal Of Usability Studied,* 4(3), pp. 114-123.

Bhardwaj, S., Sharma, R., Chouhan, P. & Sharma, P., 2013. Android Operating Systems. *International Journal of Engineering Technology & Management Research,* 1(1).

Corral, L., Succi, G. & Silitti, A., 2013. Agile Software Development Processes for Mobile Systems: Accomplishment, Evidence and Evolution. pp. 90-106.

Fakultas Ilmu Komputer, 2017. Integration Testing. Dalam: A. Hendrabrata, penyunt. *Software Testing Level.* Malang: Fakultas Ilmu Komputer, pp. 19 - 45.

Flora, H. K. & Chande, S. V., 2013. A REVIEW AND ANAYSIS ON MOBILE APPLICATION DEVELOPMENT PROCESSES USING AGILE METHODLOGIES. *International Journal of Research in Computer Science,* 3(4), pp. 9 -18.

Fowler, M., 2003. *UML Distilled Third Edition A Brief Guide to the Standart Object Modelling Language.* 3th penyunt. Boston: Addison Wesley.

KhedKar, S. & Thube, S., 2017. Real Time Databases for Application. *International Research Journal of Engineering and Technology,* 04(06), pp. 2078 - 2082.

Leau, Y. B., Loo, W. K., Tham, W. Y. & Tan, F. S., 2012. Software Development Life Cycle AGILE vs Traditional Approaches. *International Conference on Information and Network Technology,* Volume 37, pp. 162 - 167.

Myers, G. J., 2004. *The Art of Software Testing, Second Edition.* Second Edition penyunt. Canada: John Wiley & Sons, Inc., Hoboken, New Jersey.

Nielsen, J., 2000. *Why You Only Need to Test with 5 Users.* [Online] Available at: https://www.nngroup.com/articles/why-you-only-need-to-test-

with-5-users/
[Diakses 22 12 2017].

PERMENKOMINFO, 2017. *eraturan Menteri Komunikasi dan Informatika Nomor 12 Tahun 2016 tanggal 4 Agustus 2016.* [Online] Available at: https://jdih.kominfo.go.id/produk_hukum/view/id/541/t/peraturan+menteri+komunikasi+dan+informatika++nomor+12+tahun+2016+tanggal+4++agustus+2016 [Diakses 28 Oktober 2017].

Perry, W. E., 2006. *Effective Methods for Software Testing.* 3th penyunt. Indianapolis, Indiana: Wiley Publishing, Inc..

Pressman, R. S., 2010. *Software engineering: a practitioner's approach.* 7th penyunt. s.l.:McGraw-Hill.

Rambaugh, J., Jacobson, I. & Booch, G., 2005. *The Unified Modeling Language Reference Manual.* 2nd penyunt. Canada: Addison Wesley.

Sapataru, A. C., 2010. *Agile Development Methods for Mobile Applications.* UK: University of Edinburgh.

Sharfina, Z. & Santoso, H. B., 2016. An Indonesian Adaptation of the System Usability Scale (SUS). *ICACSIS,* pp. 145 - 148.

Softwaretestingfundamentals, 2016. *ACCEPTANCE TESTING Fundamentals.* [Online]
Available at: http://softwaretestingfundamentals.com/acceptance-testing/ [Diakses 17 Maret 2017].

Sommerville, I., 2011. *Software engineering.* 9th penyunt. s.l.:Addison-Wesley.

STAPIĆ, Z., 2013. *A PROPOSAL OF AN ONTOLOGY-BASED METHODOLOGICAL FRAMEWORK FOR MULTI-PLATFORM MOBILE APPLICATIONS DEVELOPMENT.* Alcalá de Henares (Madrid): University of Alcalá Computer Science Department, Postgraduate School Doctoral program "Information and Knowledge Engineering".

Statcounter, 2017. *Mobile Operating System Market Share in Indonesia.* [Online] Available at: http://gs.statcounter.com/os-market-share/mobile/indonesia/#monthly-201603-201703-bar
[Diakses 4 April 2017].

Supriyono, H., Saputra, A. N. & Sudarmilah, E., 2014. RANCANG BANGUN APLIKASI PEMBELAJARAN HADIS UNTUK PERANGKAT MOBILE BERBASIS ANDROID. *JURNAL INFORMATIKA,* Volume 8.

VTT Electronics, 2004. *Products & services Mobile-D.* [Online] Available at: http://agile.vtt.fi/prodserv.html [Diakses 14 Maret 2017].

WeareSocial, 2017. *Digital in 2017: Global Overview.* [Online] Available at: https://wearesocial.com/sg/blog/2017/01/digital-in-2017-global-

overview
[Diakses 28 Februari 2017].